Communally Governed Transactions among Collaborative and Decentralized Trading Agents

Avinanta Tarigan

Avinanta Tarigan
AG Rechnernetze und verteilte Systeme (RVS)
Technische Fakultät
Universität Bielefeld
email: avinanta@rvs.uni-bielefeld.de

Abdruck der genehmigten Dissertation
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Technischen Fakultät
der Universität Bielefeld
am 20. Juli 2007 vorgelegt von Avinanta Tarigan,
am 12. November 2007 verteidigt und genehmigt.

Gutachter:
Prof. Peter B. Ladkin, Ph.D., Universität Bielefeld
Prof. Harold Thimbleby, Ph.D., University of Swansea

Promotionauschuß:
Prof. Peter B. Ladkin, Ph.D., Universität Bielefeld
Prof. Harold Thimbleby, Ph.D., University of Swansea
Prof. Dr. Ipke Wachsmuth, Universität Bielefeld
Dr. Karsten Loer, R&D - Germanischer Lloyd AG

Gedruckt auf alterungsbeständigem Papier ISO 9706

Communally Governed Transactions among Collaborative and Decentralized Trading Agents

Dissertation zur Erlangung des Grades eines Doktors der
Naturwissenchaften (Dr. rer. nat.)
der Technische Fakultät der Universität Bielefeld

vorgelegt von
Avinanta Tarigan

2007
(Tag der Disputation 12.11.2007)

© 2007
Avinanta Tarigan
All Rights Reserved

to my mother, my wife, my son,
and to the loving memory of my father

untuk yang tercinta
istriku Hertati, anakku Rafie,
mamaku Isminarti,
dan Almarhum Papa
Tambin Tarigan

Acknowledgments

All praise are due to God the almighty, the most gracious and the most merciful. Without His Consent this work would not come into real.

I started this work with tiny confidence it is a worthwhile research. The completion of this thesis is nothing more than tremendous support from the people whom I would like to thank.

Firstly, I would like to express my grateful to my supervisor Prof. Peter B. Ladkin, Ph.D. Without his comments, continuous support, and fruitful discussions, this thesis would be another bunch of papers. Secondly, I would like thank Prof. Harold Thimbleby, Ph.D., who were willing to allocate his time for being the second reviewer of my thesis. Thirdly, I very much acknowledge Prof. Dr. Ipke Wachsmuth and Dr. Karsten Loer for being the member of my Ph.D. committee. Furthermore, I thank the Technische Fakultät for their support in the administration, foremost, Prof. Dr. Jens Stoye and Anke Weinberger.

This work has been funded by a scholarship from TPSDP Project and Gunadarma University. I thank them for giving me the opportunity to undertake my Ph.D. I grateful to Prof. Dr. E.S. Margianti and Prof. Suryadi Harmanto, S.Si., MM., as well as the big family of Gunadarma University for their unbelievable support and understanding.

My family is the best thing that ever happens to me. I grateful to my mother and my beloved wife Tathie for their continuous support, pray, and patience helping me toward the finalization of my work. Of course, my son Rafie who energized me with his funny smile. The fact that I have been leaving them in Indonesia for almost three years, makes their patience and sacrifice absolutely unpaidable.

Last but not least, I would like to express my gratitude to all friends during my stay in Bielefeld: the family of Wiryana, Abdurrouf, Terima, Nugroho, Mormann, the RVS Guys: Heiko, Andreas, Mirco; and all Indonesian friendships in Bielefeld.

This page is dedicated to all beloved persons who have supported me directly and indirectly.

Statement

The work in this thesis is based on research carried out at the Networks and distributed System working group (AG RVS), Faculty of Technology, University of Bielefeld. To the best of my knowledge, no part of this thesis has been submitted elsewhere for any other degree or qualification and it's all my own work unless referenced to the contrary in the text.

Abstract

This thesis proposed a framework in completing transactions among decentralized agents without the existence of trusted authority nor intermediate facilitator in facilitating the transactions. The framework consists of transaction concept and logic, set of algorithms to accomplish transactions, and architecture that enable agents to govern and self-organize accounting, authentication, and authorization of transactions for mutual benefit. The system of transaction is based on the concept of creation and elimination of institutional facts as the fundament to develop the transaction logic. Set of agents united in a trading community collectively accept set of arbitrary constructed assertion of facts establishing the presence of accounts of finances of transacting agents. During transactions, those accounts are altered through set of collaborative actions that agents perform specified by transaction algorithm. This social-based accounting mechanism introduces a mechanism that enables agents to collectively govern transactions in individual level. By considering reputation of the seller as well as the ratio between potential risk and benefit of every transaction, agents perform vote-based collective decision to authorize or not to authorize the transaction. The feedback of each transaction is made public altering reputation of the corresponding seller. The collective authorization mechanism establishes social control to block possible bad behavior, to protect buyer from risky transaction, and to induce good behavior. The result of simulation on this scheme shows how such social control works. It filters bad behaving agents and preserves good behaving agents. This condition is necessary to sustain trust within community and thus preserves the collaboration. A chapter of this thesis presents set of protocols developed from transaction algorithm that fulfills best-practice considerations. It successfully applies distributed cryptography to devise mutual authentication problem and to reduce communication cost as well as to eliminate scalability issue of original algorithm. Finally, an application of the framework is exemplified: an architecture of decentralized trading network that superimposes existing P2P networks allowing Internet users to trade for their mutual benefit.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviation

TTP	Trusted Third Party
CA	Certification Authority
PKI	Public Key Infrastructure
P2P	Peer-to-Peer Network
CF	Collaborative Filtering Systesm
CS	Collaborative Sanctioning System
MNR	Multiparty Non-Repudiation
UP	User-side Program
DP	Delegation Program

Chapter 1

Introduction

1.1 Preface

The decentralization nature of Internet opens many new spaces of research in finding ways to conduct sustainable trading on the network. This research was started with a wish to develop a framework in completing digital transactions that can be used in decentralized system where trusted third parties (TTP) is absent or socially very weak. The aim is to develop trading infrastructure for Internet based decentralized system such as Peer-to-Peer (P2P) network.

In nowadays Internet Commerce, digital-transactions are performed using algorithms that assume the existance of mutually trusted third party administering the transactions. Figure 1.1 illustrates the ideal condition where Internet users may perform digital transactions securely. Government agencies, e.g. Chamber of Commerce, administers professional examination to examine trustworthiness and *bona fide* of trading entities. The national ID program provides secure and effective identification system of all citizens. Certification Authority (CA) issues or revokes digital-certificates for the parties who receive authorization from the Government to take part in Internet commerce. During transaction, transacting parties are mutually authenticated using trusted digital-certificates whilst bank and payment gateway are carrying out the accounting as well as financial settlement. Government is responsible to monitor and control the trading parties through feedback mechanism in order to maintain fair trading as well as trust, adhering sustainable transactions.

This illustration exposes duties of TTP to help traders in achieving transaction objectives or in accomplishing properties that are necessary for a proper trading infrastructure, namely, to establish trust in authentication, to provide accounting and financial settlement services, as well as to authorize transactions sustaining trust among transacting parties.

In certain circumstances things do not run ideally, however. In particular, the are conditions where TTP is socially very weak or completely absent. For example,

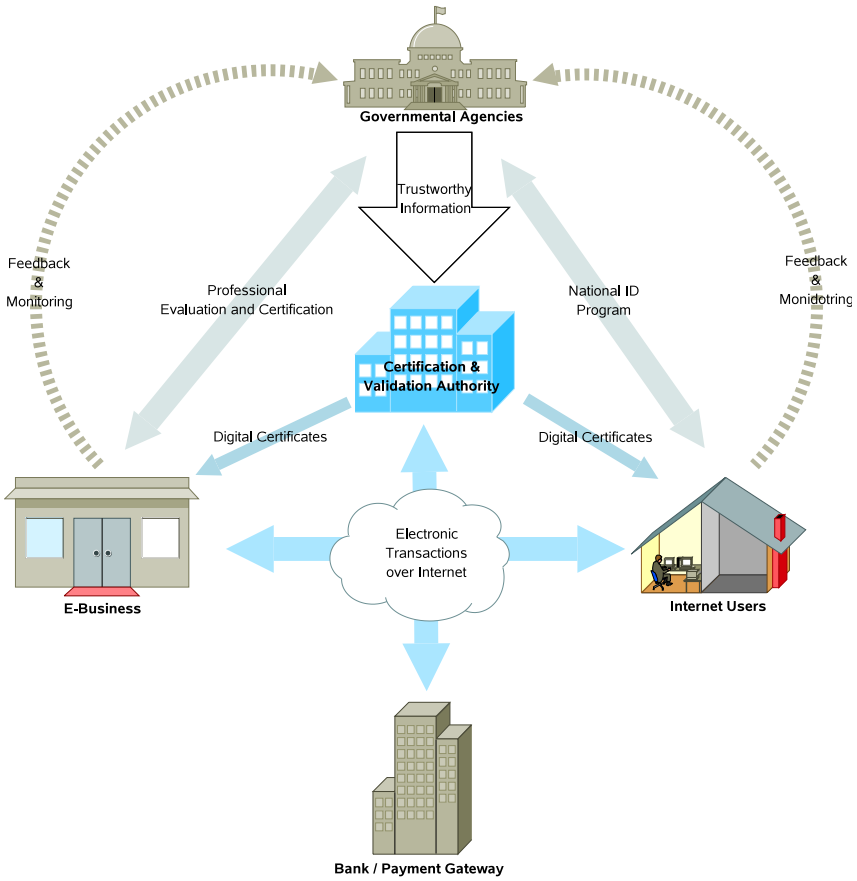


Figure 1.1: The completion of digital transactions using TTP

how can one trust someone being certified by a government whom people does not trust? How can one ensure that a digital-signature is authentic when the issuer of correspondence digital-certificate is unknown or resides outside one's jurisdiction and control?

Some Internet applications, such as P2P file-sharing networks, simply avoid the presence of TTP. Despite that lots of Internet users utilize P2P networks to exchange and share their data, the system lacks from properties of proper trading infratructure. That is, the desired feature of being uncontrollable forbids the development of P2P from free network to proper trading network on which P2P users are able to trade their resources in adequate manner. Some of solutions surveyed in this thesis are based on the idea of giving incentives, namely, to balance utilization of peers with their contributions. But not to have proper trading activities that are close to common trading activities in ideal condition.

To sum up, it is worthwhile to research methods that enable digital transactions in decentralized network. Since the demand is considered to be high and the

remaining research spaces in this area are still vacant.

1.2 Contributions

The main focus of thesis is to investigate ways to establish desired properties of proper trading infrastructure in the decentralized system, namely, providing accounting service, establishing authentication, as well as maintaining trust among users without any assistance from TTP. The search leads the effort to develop scheme that promotes collaboration in order to establish those properties. The first objective is to design transaction logic and algorithm as the ground work that will be further developed to transaction protocol that is feasible to work in decentralized system.

The contributions of this thesis which are considered beneficial for the research in this area are:

- The formalization of transaction logic based on the concept of institutional facts brought from the theory of the construction of social reality. The logic provides the basis for the concept of the presence of accounts of finances, as well as the basis to develop transaction algorithm that enables self-organized accounting scheme.

- The design of collective authorization scheme reasoned by reputation that establishes social control. The scheme allows agents to maintain trust by promoting good behavior and discouraging bad behavior.

- The design of transaction protocols applying distributed cryptography in order to establish group authentication as well as to accomplish secure transactions.

- In addition, the architecture of a trading infrastructure for P2P network, in that proposed framework is applied, is presented in the final chapter. It exemplifies the implementation of proposed framework in the real use.

Furthermore, part of the work described in this thesis is published in the following paper:

[Tarigan, 2006] Tarigan, Avinanta. Towards Communal Governed Transactions Among Decentralized Trading Agents. In *Proceedings of The Second International IEEE Workshop on Security Through Collaboration (SECOVAL) 2006*, Conference on Security and Privacy in Communication Networks (SECURECOMM 2006).

1.3 Thesis Overview

The following chapter reviews the relevant literatures and concepts providing background on which this thesis is developed. Some of the related works in the same research area are also detailed here.

Chapter 3 develops the formalization of transaction logic. It shows how transactions can be conducted without the existence of TTP. The transaction algorithm developed from the logic is specified using Temporal Logic Action (TLA+) in order to explicitely express collaborative actions.

The subsequent chapter develops collective authorization scheme with a wish to accomplish social control within the system of agents. Here, the concept of trust and reputation system is reviewed as well as community trust decision model as the basis of the authorization. At the end, the algorithm is specified in detail using TLA+ which completes the specification of transaction algorithm developed in the previous chapter.

In oder to study how the collective authorization works, the followed chapter presents a simulator in that collective algorithm is peformed on set of random data representing trading environment. Some experiments are also conducted to study behaviour of the system given predefined range of combinations of parameters. It is regarded beneficial for considerations in the future implementation of the framework.

Chapter 6 takes the algorithm one step closer into reality by presenting set of transaction protocols developed from original transaction algorithm introduced in chapter 3. The protocol incorporates distributed cryptography to establish group authentication as well as secure transactions. The achievement is that it significantly reduces communication cost as well as demotes scability problem of original transaction algorithm.

Chapter 7 adds the thesis with the design of a trading infrastructure applying the proposed framework that enables P2P users to transact. The design solves several issues raised because of the nature of P2P networks. It resolves the integration problem with existing P2P architecture as well.

The final chapter summarizes contributions and the conclusion as well as outlines potential future works emerged by the presence of this thesis.

Chapter 2

Backgrounds and Related Works

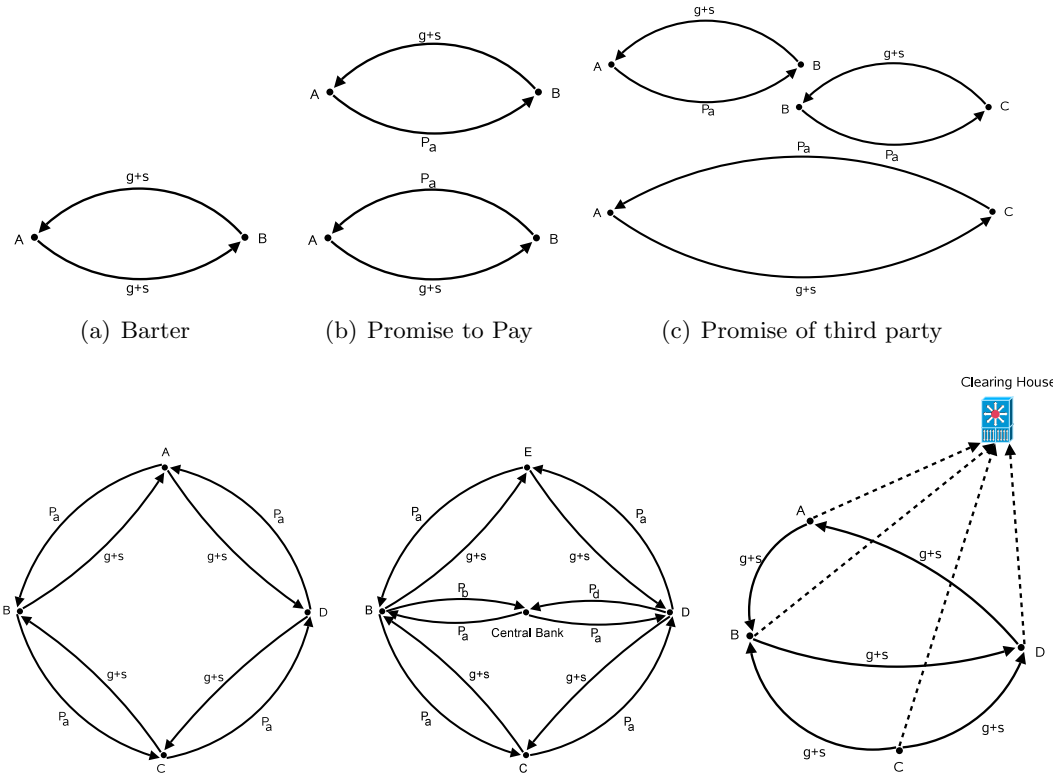
This chapter presents the backgrounds on which the framework in thesis is developed. It begins by overviewing the structures of exchange that gives comprehensive view of mechanisms of trading. The next part presents state of the art of online transaction protocol by surveying several related works including Ad-Hoc protocols and collaboration based protocols. Finally, the emerging use of the notions of interpersonal trust in online environment is briefly reviewed.

2.1 The System of Exchange

The term transaction is mainly associated with economic activity in buying and selling goods. The Merriam-Webster Dictionary [Webster, 2005] describes transaction as "*a communicative action or activity involving two parties or things that reciprocally affect or influence each other as an exchange or transfer of goods, services, or funds*". The Wikipedia Online Encyclopedia [Wikipedia, 2005] describes transaction as "*a change in the status of the finances of two or more businesses or individual*". Transaction consists of reciprocal actions in exchanging resources, resulting the change of status of each individual involved in the transaction.

Coleman [Coleman, 1990, p132] explains general framework of exchange in social system. The system of exchange consists of **actors** and **resources** or **events**. Resources are usually associated with, but not restricted to, goods or physical objects, and events with services by which the desired event occurs given the service.

Every actor has **interest** on set of resources as well as **control** over set of resources. The system begins with the initial distributions of control of actors over resources. Every actor pursues resources he interests to which might be under the control of other actors. When actors meet each other and decide that the exchange



(a) Barter (b) Promise to Pay (c) Promise of third party

(d) Ringtausch (e) Promise of Central Bank (f) Cashless System

Figure 2.1: Structures of Exchange [Coleman, 1990, p.122-125] resources $\{g + s\}$, promises $\{P_a, P_b, P_d\}$

of control of resources is best to satisfy them, then the transaction occurs. That is, one actor gives his control of the resources to his transacting partner(s) and *vice versa*. This reciprocal action is called exchange.

2.1.1 Structures of Exchange

Coleman describes the structures of exchange, presented in figure 2.1, which have been practiced within social system. The oldest and known mechanism is so-called *Barter* exchange initiated by two or more actors who coincidentally have interest in each others' resources. Figure 2.1(a) illustrates the barter exchange in that two actors reciprocally exchange their control over the resources at the same time.

However, since the chance of *a double coincidence of wants* is likely rare to occur, human beings developed mechanisms to break apart the halves of the double coincidence of barter transaction. Figure 2.1(b) illustrates a mechanism in which one actor gives his control of resource to his partner, but the partner delivers him with a promise to pay. Here, one actor has interest to the resources of other actor,

but the corresponding actor does not have yet the reciprocal interest. The pay-back takes place later when the debiting actor has interest on the resource of owning actor.

The promise-to-pay mechanism can also be accomplished by three actors. For example, A has interest on the resource belonged to C, but B wants what A has, and C wants what A has. Figure 2.1(c) shows how to accomplish the transaction. First, A gives B his word that A will pay for resource that B delivers to him. Next, B hands over $A's$ word to C for the resources that C gives to B. And at the end, C claims A's promise to be exchanged with resources that C wants from A.

Figure 2.1(d) illustrates a more extended mechanism so-called *Ringtausch*. The structure of promise-to-pay is extended that it forms a ring of exchange. In this ring of exchange, the word of promise-to-pay is passed from one actor to another, to be exchanged of resources, until it comes back to the one who issued it.

The extention from promise-to-pay mechanism is what has been conducted in today's transaction system. Figure 2.1(e) shows the cash system in which central bank has important role as the single trusted third-party issuing promise-to-pay token which is used as medium of exchange within the domain of the country where the central bank is the authority of the economy system.

The modern way to transact is shown in figure 2.1(f) where trusted clearing house performs exchange settlement process. In cashless system, bits of information stored in the clearing house represent the financial status of the actors. When actors perform transaction, the process is reported to the clearing house by which their records are changed according to the amount of transactions.

2.1.2 Money

The promise-to-pay mechanism described above has been developed into the state where medium of exchange known as money is used to represent the promise-to-pay token issued by the government. At the beginning, people used valuable material such as silver and gold which are known as *commodity money*. Here, the prices were standardized to the value of the material used. However, people found that it was difficult to trade using gold or silver due to the weight and the form. Moreover, the properties of the material such as pure gold can be very weak in certain circumstances.

In order to accommodate this issues, the government, through central bank, issued *fiduciary money,* tokens made from non-valuable but relative robust material, such as metal or paper. In this monetary system, the total value of issued fiduciary money should be the same as the total value of gold stored in the central bank. That is, the gold stored is the guarantee of promise-to-pay by the central bank.

Nowday's monetary system uses *fiat money* which is less than such promise. It is declaration from the government that the currency is legal tender for all debts

within the domain of the government. Thus, fiat money is accepted because government legalizes its presence and guarantees the continuity in using that money in the domain of the government.

2.1.3 Some Important Remarks

Despite the limitation that the probability of double coincidence of wants is low, one can take advantage of Barter exchange from its simplicity. Assuming the agreement is already established and fair for both sides, then the transaction is relatively easy to conduct. Accomplishing transaction requires only the transacting actors to be involved in the exchange and no further actions or conditions should be concerned after the completion. Hence, Barter exchange does not require that the actors knew each other prior to transaction. That is, transaction can be instantly happened even by strangers.

On the other hand, promises-to-pay inhibits interdependent relations among actors. One actor depends on the word of other that he will pay back the debt. It also introduces time lag between the investment, as the debiting actor hands over his control over resources to the owing actor, and the completion of the transaction, as the owing actor fulfill his promise. Indeed, this mechanism introduces the risk and thus requires trust to be established before transaction takes place.

With respect to Barter exchange, one might find that the using of money as medium of exchange exhibits the same properties of Barter exchange. Of course actors can buy resources using money without previously knowing each other. In fact, the lack of trust can also be eliminated, since one could previously examine the goods before buying. However, one tends to forget the role of government that guarantees the continuity in using the money. Thus, instantaneous transaction using money is possible because there is underlying social and economic infrastructure, which in this case is performed by government, that might be transparent to the traders.

2.2 Transactions in Distributed Systems

Generally, the term electronic transactions is used to describe the actions in exchanging information among distributed and networked computers that affect each other's states. In order to reach the objective of particular transaction, these actions are specified and regulated by set of protocols or algorithms.

Database transaction, for example, is the actions implicating the change of state of the objects or records managed by a database server [Coulouris et al., 2001]. The objective of the protocol is to ensure that all objects remain in consistent state given multiple transactions or concurrency access. The two-phase-commit protocol is the example of transaction algorithm used in database system. It brings the parties from

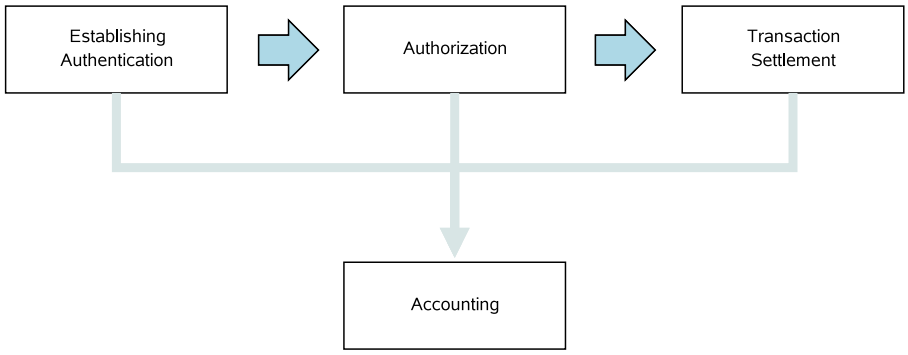


Figure 2.2: Typical Transaction Stages

initial states of transaction to the synchronized final states: whether the requested transaction is committed or aborted. Here, the database server is the central entity that manages all transactions and connections made by database clients, as well as performs access control to every objects stored in the database.

As exemplified in the description above, most of the transaction protocols in distributed systems utilize central entity that administers transactions. It simplifies the interactions needed to reach the objective of the transaction and helps in the design of the system to eliminate particular constrains which can be solved utilizing central entity. However, it introduces single point of failure such that the failure of central entity in delivering its functions would affect the whole system.

Furthermore, conducting transaction over open and distributed network infrastructure raises some issues namely how to accomplish authentication, authorization, as well as accountability. Figure 2.2 illustrates typical electronic transaction scheme where authentication and authorization stages has to be done before the intended transaction settlement can be carried out. Here, every important change in each stage should be recorded using accounting mechanism to promote accountability. The following descriptions sketch techniques in accomplishing authentication, authorization, and accountability which have been used in digital transactions.

2.2.1 Authentication

One of the major problems in conducting transactions open and distributed computer systems is the lack of message authenticity. Messages transmitted in open network can be produced, manipulated, and reproduced by any node connected to the network. Therefore, the authentication protocols are designed to establish the properties of authenticity, integrity, and freshness of messages [Anderson, 2001]. Thus, the transaction of authentication brings communicating parties to the state that every party believes that it talks with genuine one.

The modern authentication protocol incorporates cryptographic algorithms to

establish security properties in which cryptographic keys are used to verify credentials. The keying materials and credentials are stored and managed by central entity. Here, the central entity is called trusted entity, thereby the parties should trust the entity to administer the authentication sessions or to deliver the valid authentication related information. The failure of trusted entity would break the security of the system. Therefore, one of the objective of the research in this area is to design trustworthy system, the system that never fails in delivering its security functions.

Some of the authentication protocols which have been used in distributed system are the Needham-Schroeder Protocol [Needham and Schroeder, 1978], the Kerberos Protocol [Steiner et al., 1988], and their derivatives. These protocols require the existence of trusted entity to manage keys and to establish authentication session among principals. The X.509 Public-Key Infrastructure framework, which was originated from MIT Bachelor Thesis in 1978 [Kohnfelder, 1978], incorporates trusted-third-party so-called Certification Authority to administer digital-certificates. Despite the fact that some have criticized its implementation [Ellison and Schneier, 2000], the X.509 digital-certificates have been widely used for authenticating web sites and emails, as well as providing tools for legal infrastructure of Internet commerce.

2.2.2 Authorization

After authentication is established in which credentials are verified and validated, a principal should determine into which states its system are allowed to engage given access from authenticated party and particular circumstances; such as time and location. That is, authorization is about the rules of what are allowed and what are not allowed in the system. System policy defines these rules and access control mechanism enforces them.

In security perspective, authorization is about capabilities or permissions for the accessors, in accessing objects under authority of the system. The permission structure is modeled in access control model such as Multi Level Security (MLS) used in military information system, Discretionary Access Control (DAC) which have been implemented in file system permission on UNIX operating system, and Role Based Access Control (RBAC) which gives one more flexibility to express security policy.

In a broader perspective, authorization is about all means in sustaining the system behavior to conform with the system policy. In this context, system policy concerns not only security policy, but also general policy such as business policy and organization policy. For example, an online credit-card based payment system is designed that it could only authorize transactions with worth under 1000 Euro for silver member, or an SMTP server in an organization should reject incoming e-mail whose size is over 50 Megabytes, etc.

In distributed systems, every party enforces the rules locally. However, in large

systems these rules are so complex and dynamic that it should be managed by the trusted entity. The parties retrieve authorization related information from the trusted entity and use these information as parameters in deciding whether to authorize or not to authorize the access.

Trust Management

Modern authorization and authentication for distributed systems requires general framework which has more flexibility in expressing security policy. *Trust Management* introduced by Blaze et al. [Blaze et al., 1996] offers an unified approach in specifying and interpreting security policies, credentials, and relationships. In their paper, Blaze et al. proposed the first Trust Management entitled with *PolicyMaker* in which credentials and policies are referred to as assertions. These assertions can be programmed in any programming languages from which flexibility is gained. Moreover, compliance proofs, that all policies are satisfactory and compliance to each other, can be conducted through repeated evaluation of those assertions. In other words, the language enables policy verification using model checking. The *KeyNote* [Blaze et al., 1999a] Trust Management was developed as enhancement of PolicyMaker in which authors aims to include standardization of policy and to design the framework that is easy to be integrated into applications.

Some other trust managements are *Referee* [Chu et al., 1997], *Fidelis* [Yao, 2003], *Cassandra* [Becker and Sewell, 2004], *RT* [Li and Mitchell, 2003], and *Sultan [Grandison and Sloman, 2001].* Each trust management is designed to achieve special features. Sultan, for example, is trust management specializing on reputation based authorization. The policy language in Sultan is designed to include reputation information of each node according to managed reputation system.

Note that the Trust Management does not directly enforce those policies [Blaze et al., 1999b]. It works like trusted reference for the whole network. Making sure that every party complies with the global policy is still unanswered.

2.2.3 Accounting and Accountability

In digital transaction, accountability is about function and capability of the system to record and track important changes caused by one's action and use it as the proof. It should answer at least the questions of who, what, and when, regarding the change. In the implementation the recording function of accountability is implemented using accounting system.

In the context of security, accountability is focused on all means in knowing, recording, and proving who is responsible for an action causing the change in the system. Accounting mechanism should able to record authentication and authorization sessions, as well as any security related information regarding the change. Good

accounting should promote the state of non-repudiation, that party whose action has changed the system won't be able to deny his action.

Nevertheless, accounting system is not to be used only for accountability in the context of security. Accounting mechanism in database system is designed that one should be able to store the audit trails of the transactions in order to have capability to replay-back the transactions when the database crashes or inconsistent.

In e-commerce, accounting is mandatory in order to record changes of financial status of the parties performing business transactions. Here, accounting system is the core system providing centralized electronic accountancy system in distributed system. Accounting agency is usually performed by the Bank or authorized payment gateway.

An operating system provides accounting service for its applications. The accounting information are stored in log files which are protected in order to maintain their integrity. The Unix operating system, for example, provides so-called *syslog* daemon to provide logging service to the other process. The logs are stored in */var/log* directory which is owned by the *root*. In networked systems, it is necessary to store the accounting related information in an accounting server. The Internet Service Provider (ISP), for example, uses *Radius* server to store all connections made by the customers for billing and auditing purposes. In order to maintain integrity, accounting database and log files are periodically backup-ed and digitally-signed before being stored in the protected media.

2.2.4 The Roles of TTP in Accomplishing Transactions

Previous description gives a perspective of the roles of TTP is in completing the transactions. The following passages describes common roles of TTP:

- Establishing Trust
  Without doubt, trust is adherence of collaboration. TTP provides establishment of trust in such way that TTP guarantees that the party whom it trusts can also be trusted by other parties. Here, trust relations establishes in hierarchical form where TTP is located at the root of the tree. Thus, by trusting the root of trust, parties located in the leaves of the hierarchy are able to trust each other.

- Guarantee Fairness
  In an exchange, it is necessary to have final state where all transacting parties get the resources according to the agreement. In order to simplify the mechanism a TTP is incorporated in the system to facilitate the exchange. The role of TTP is to re-balance unfairness for the party that hands over his resources first [Nenadic, 2005]. For example, TTP collects resources from all parties, verifies them, and ensures their deliverability to the destined parties.

- Enforce Non-Repudiation
  Some transactions require the state where no party can deny the action it has made. This property is called non-repudiation which in real life is performed through legal witnesses or using signatures. Here, TTP witnesses important actions being done in the transactions and performs its role as agency of accountability. In the PKI, the TTP is performed by Certification Authority or Validation Authority providing notary services to its customers.

- Legal Agency of Accountancy
  Electronic payment systems needs centralized scheme to provide core business transactions with legal accounting agency. TTP, which is usually performed by the Bank, manages member's accounts and responsible of all changes during business transactions.

2.3 Transactions without Trusted Third Party

Previous description illustrated how important the TTP is in completing transactions. This section, however, discusses and reviews some approaches in which the usage of TTP can be reduced or eliminated. There are two identified approach, first, is the use of ad-hoc protocol to solve specific problems. Second, is to promote collaboration or coordination in communities to work together in order to deliver the function of TTP.

2.3.1 Ad-Hoc Protocols

Ad-Hoc protocol is algorithm to solve specific transaction problem that does not require long-term authentication or accountability. It is mostly designed for Ad-Hoc network and has advantage that it can be used to complete instantaneous transactions among strangers. Some of the protocol is able to maintain anonymity whilst the parties carry out the transaction.

The analogy of Ad-Hoc protocol is Barter exchange. They are simple and can solve instantaneous transaction, even among complete strangers. Yet, the drawback remains, that sustainability of trust can not be achieved using the protocol. Therefore, it can be implemented to solve only special transaction problem. The following description reviews known Ad-Hoc protocols.

Gradual Releases Fair Exchange Protocols

Fair exchange protocol is the analogue of barter exchange conducted in online environment. The purpose of fair exchange is to allow two or more parties to reciprocally exchange their resources in such a way that every party gives the resource away if and only if it receives wished resource in return.

Some of fair exchange protocols are designed that they don't need TTP to balance unfairness. The typical mechanism of this protocol is that in every state one party releases a chunk of his resource to whom he is exchanging and receives a chunk of wished resource from his counterpart. This exchange is repeated in that all parties gradually receive the resources. At the final state of the protocol, all parties should receive the all chunks from that the complete exchanged resources can be constructed. Here, both parties need no third party to facilitate the exchange.

Earlier development of fair exchange protocol was concentrated on how to exchange secret [Blum, 1983, Okamoto and Ohta, 1994, Damgard, 1994]. In the first stage of the protocol, each party computes and declares his commitment of his secret using Zero-Knowledge-Proof protocol (ZN)[1]. Next, the secrets are exchanged in turn-based mechanism. In every turn, each party verifies the chunk he receives with the correspondent commitment. At the final state, every party should have the complete set of verified chunks and thus the secrets can be constructed.

Okamoto and Ohta [Okamoto and Ohta, 1994] developed the application of the gradual releases fair exchange protocol to solve the problem of exchanging digital-signatures of the contract document. In this problem, one party is given opportunity to cheat by not giving his signature after receiving his partner's signature. Here, the victim is the only one who is accountable for the contract document from which the cheater might take advantage. By conducting the protocol, each party should possess counterpart's signature at the end of the protocol.

Nonetheless, Nenadic [Nenadic, 2005] pointed out that fair exchange protocols that is based on gradual release of secret is impractical for real-life application. One of the reason is the protocol requires large number of communication costs between parties to conduct the ZN protocol. Moreover, it introduces opportunity to conduct semantic or syntactic attack to reveal the secrets having sufficient information that one already received during protocol run. That is, one stops the protocol after revealing the complete information from incomplete information being exchanged.

Key Agreement Protocols

Symmetric cryptography allows two party to secretly exchange information over public network. Yet, the problem of exchanging cryptographic keys over the network has been the problem for security experts.

Diffie and Helmann [Society, 1999] were among the first who found the first step towards key agreement protocol in that both parties are able to agree on a key without utilizing TTP. Suppose A and B are the parties who want to exchange information and need to agree on a symmetric key. First, A and B agree on a prime

[1]The Zero-Knowledge-Proof protocol is cryptographic protocol which enables one to prove to other that he owns particular information without revealing it to others. For further reading on ZN please see [Quisquater et al., 1989] [Goldwasser et al., 1989] [Ben-Or et al., 1990]

p and integer $\alpha \in \mathbb{Z}_p$. Then, A chooses random integer i_a such that $0 \leq i_a \leq p-2$ and sends $\alpha^{i_a} \, mod \, p$ to B. At the same time, B chooses random integer i_b where $0 \leq i_b \leq p-2$ and sends $\alpha^{i_b} \, mod \, p$ to A. A computes $K_{ab} = \left(\alpha^{i_b}\right)^{i_a} \, mod \, p$ and B computes $K_{ab} = \left(\alpha^{i_a}\right)^{i_b} \, mod \, p$. That is, both parties end up having the same K_{ab} which is the symmetric key used to encrypt or decrypt information transmitted between them.

Unfortunately, the simple Diffie-Helmann protocol suffers from man-in-the-middle attack. That is, the malicious party could intercept the communication and pretends that he is genuine party. Therefore, enhancement of Diffie-Helmann protocol was developed, the so-called authenticated key agreement. The examples of this protocol are Station-to-Station protocol (STS) [Diffie et al., 1992] and MTI Key agreement protocol [Matsumoto et al., 1986].

Cocaine Auction Protocol

Cocaine Auction Protocol [Stajano and Anderson, 2000] is an auction protocol that utilizes anonymous broadcast protocol to complete auction amongst anonymous parties. The purpose is that the bidding process remains anonymous but at the end seller is able to authenticate the winner of the bidding.

Let e and N be public key component of a public-key encryption algorithm. Seller starts the auction by broadcasting components of public-key e and N. In every round of the bidding, seller announces a raised price which will be answered by bidder who commits to buy on that price by broadcasting $e^{x_i} \, (mod \, N)$, where x_i is nonce chosen arbitrarily by bidder b_i in round i.

In the last round j where nobody answer the proposed price, the winner of the bidding is the one who has answered on round $j-1$ with $e^{x_{j-1}} \, mod \, N$. In order to identify w_{j-1}, the seller chooses a nonce y and performs a Diffie-Hellman Key Exchange with the winner w_{j-1} by broadcasting the appointment encrypted under the session key $e^{x_{j-1}y} \, (mod \, N)$. Thus, w_{j-1} is the only buyer who can compute this key.

During protocol run, nobody can reveal x_i from $e^{x_i} \, (mod \, N)$ except the seller himself who has the correspondent private-key. Thus, anonymity of the bidder can be protected. Stajano and Anderson named the protocol "Cocaine Auction Protocol" because this is associated with typical problem exists amongst Cocaine mafia, in which no dealer wants his identity to be revealed to others for their security against law.

2.3.2 Completing Transactions with Collaboration

This description reviews some of the approach which promotes collaboration in the absent of TTP. Most of the approaches utilize the Threshold Cryptography system

in order to jointly replace the function of CA in the PKI. The t-out-of-n threshold cryptography provides collective signaturing scheme which allows t peers out of n peers to perform the signature. Thus, availability of the peers can be preserved during the absence of $n - t$ peers and the security of the secret can be protected since the system needs at least t peers to reveal the secret.

The PGP's Web of Trust

In contrary to PKI, the Pretty Good Privacy (PGP) promotes group authentication which does not rely on Certification Authority. Instead of having hierarchical structures, the trust relations established in form of chain of trust. Each PGP user certifies or digital-signs other's certificate whom he knows or trusts establishing mesh structure of certification. Through this structures a user x can search through the chain to find out whether user y is trusted, e.g. belongs to the x chains of trust. This process can be explained by the following example.

Let $X << Y >>$ represents that X certifies PGP certificate of Y. A user Bobby needs to communicate with Alice. In order to trust this certificate Bobby should find chain which links his certificate to Alice's certificate. Bobby obtains Alice's PGP certificates and finds out that $Peter, Petra, Mormann << Alice >>$. Next, he obtains all certificates of Alice's certificate certifier and finds out that $Zara, Tabitha << Mormann >>$. During the search, Bobby finds a final link that $Bobby, Shara << Tabitha >>$. Thus, he can trust Alice's certificate from the chain:

$$Bobby << Tabitha >> \rightarrow Tabitha << Mormann >> \rightarrow Mormann << Alice >>$$

Group Authentication among Mobile Devices

Establishing authentication in decentralized network is difficult to accomplish since the nodes come and go and anonymity should be place on the first priority. Nevertheless, in order to conduct continuous relationships among peers in the sense of transaction, authentication is the key. Quercia et al. [Quercia et al., 2004] propose an authentication framework for decentralized network which offers unique identification, off-line authentication, and non-repudiation, but still maintains anonymity.

The key to establish such framework lays on the extensive use of blind t-out-of-n threshold signatures algorithm. During group bootstrapping, the members jointly generate group public-key as well as n secrets using protocol proposed by [Boneh and Franklin, 2001] and jointly certify each other's public-key using t-out-of-n threshold signatures. In order to prevent disposable pseudonyms, each prospective member should run through 5-steps induction protocol from which he should get responses from the quorum t members in form of his certificate as well as receiving his part of secrets to be used in joint t-out-of-n signatures. Using this scheme, two member can

authenticate each other directly using their certificates without using online services of TTP.

Incentive Based Accounting for P2P Network

"Free-riders" problem has been known as major problem in P2P system: peers that consume other's resources more than their fair share of a resource. In order to devise this problem, some have developed solutions that are based on coordination among peers. There are two approaches developed to overcome this issue [Androutsellis-Theotokis and Spinellis, 2004]. The first is based on the reputation system that consists of collaboration procedures that advises peers on the information of frequent uploaders and free-riders. Based on the advisory every peer can select who can download the files from it and who doesn't. Eigentrust [Sepandar D. Kamvar, 2003, Li et al., 2005] reputation system is one of the effort belongs to this approach.

The second approach is to develop mechanism that balances the traffic between upload and download. However, there are very few that bases their solutions not to use TTP. Some of them are Yu and Singh [Yu and Singh, 2003] and Vishnumurthy et al. [Vishnumurthy et al., 2003]. Yu and Singh introduces model of dynamic pricing and micro-payment system that based on referral system. Vishnumurthy et al. develops an economic framework so-called Karma. It is based on coordination protocol performed by set of peers called bank-set that keeps track of accountability in the system. Each peer refer to the bank-set to authorize the request of resources from others.

Token Based Accounting for P2P Network

There are very few system for decentralized system, i.e. P2P, designed to deliver the function of accounting that close to day-to-day trading system. One of them is proposed by Liebau et. al. [Liebau et al., 2004, 2005] which is based on coordination among the peers. This system presumes that authentication is already setup, i.e. by a CA or group authentication described above, and that reputation system is setup and has been working to identify trustworthy peers.

The coordination system splits the community into a group of trusted peers, a group of account holders, and the rest are the transacting peers. At the beginning the trusted peers generates t-out-of-n threshold secrets in order to jointly sign the tokens. The account holders manages all peers accounts which is propagated using Distributed Hash Table (DHT). Both groups are composed from selected trustworthy peers identified using reputation system.

In this system, a token holds the name of the owner as well as the account ID that peer has in the account list. To obtain new tokens, the transacting peer locates one of the trusted peer and send the collected foreign tokens which he already got during

previous transactions. The trusted peer checks the tokens against the account of transacting peer maintained by account holders. After the checking, the requested trusted peer locates the other trusted peers in the group and sends the unsigned new tokens. The quorum t-out-of-n of trusted peers jointly sign new tokens and send them to transacting peer. The transacting peer combines all partial signatures and thus new signed tokens are obtained.

In order to conduct the transactions, the peer who acts as service consumer signs the token and sends it to the service provider peers. The service provider validates the token against double spending through the account holders. After transaction, the account of service consumer should decrease and one from service provider should increase according to the token spent. The service provider can obtain the collected tokens by performing the algorithm mentioned in the paragraph above.

2.4 Trust in Computer Mediated Transactions

2.4.1 The Notion of Trust in Computer Security

Internet, to which the world of online transactions is referred, weakens particular properties that one finds in physical interactions. The clear example is the lack of authentication which was discussed earlier. In this respect, information security offers solutions to overcome these problems in which the objective is to establish trust.

In the context of information security, trust refers to the establishment of security properties in the system, namely authentication, secrecy, availability, as well as non-repudiation. The establishment of these properties emerges trust infrastructure allowing Internet users to conduct various applications. E-commerce, for example, is considered to be the killer application of Internet that has high degree of dependency on the trust infrastructure.

The Problem

Recently, the PKI has been supporting the establishment of trust infrastructure for electronic transactions making Internet as considerably a safe place for doing business. Nevertheless, it is not sufficient to completely substitute the all properties that one finds in physical interaction. An Internet-shopper, for instance, can not directly examine quality of the goods sold at an online-shop. Nor he can not judge the performance and the after sale service of the shop, eventhough the online-shop's web server is secured using certified SSL and all e-papers are digitally-signed. It is common that the Internet-shopper should pay the goods he buys in advance before he finally receives them. Here, Internet-shopper is taking the risk that he might receive bad quality of goods or poor after sale service.

Similarly, the fair exchange protocol provides a method to guarantee the deliverability of exchanged resources. Though, it can not guarantee how long does it take for a party to deliver it, or to make sure that one party does not cheat delivering information which is contextually not worthy.

It is clear that conducting online transactions imposes the *risk* to the party who should rely on other's actions. One should be willing to trust his partner in order to achieve mutual goal. Lack of trust presents obstacle as one might waste his time as well as resources to build protection measures [Josang et al., 2005]. Thus, one of the agenda in development of Internet trust infrastructure is to incorporate interpersonal trust, the notion of trust found in social relationships, into the framework.

2.4.2 Interpersonal Trust

Trust is the notion that one often finds in engaging social relationships with others. Trust is important for human beings that without one there exists no society. Therefore, sociologists have been studying this notion quite extensively.

In the study of trust, the "first sociology"[2] considers the role of trust as beneficial for society as a whole. Misztal [Misztal, 1996], who reviewed sociological literatures of trust, identified the first two functions of trust in this area, namely (i) trust as having *integrative function* to bring social order in society and (ii) trust in playing its role as *reduction of complexity*. However, the "first sociology" refuse the study of trust to be reduced below the level of social system and thus fail to explain the notion of trust in individual level [Buskens, 1999].

In a cooperation, it is common that one should trust by relying himself to other in order to achieve mutual goal. Thus, unwillingness caused by distrust presents obstacles in the cooperation. This kind of trust is the notion that one founds in interpersonal relationships on which the "second sociology" emphasizes the explanation of the third function of trust (iii) as *lubricant for cooperation*. This study of trust is closely related with the investigation of trust in online transactions.

2.4.3 Interpersonal Trust in Computer Mediated Transactions

Consider the case of Internet shopping above, the lack of physical interaction places the Internet-shopper in uncertainty, whether online-shop will perform accordingly to what he expects. Moreover, it is possible that the both parties are not located under the same jurisdiction of law. Hence, legal contract can not be established implying that online-shopper has no control on online-shop. Presented with uncertainty and

[2] Sztompka [Stompzka, 1999] considers two mainstream of sociology: the *first sociology* focuses on "social organisms" - the system as a whole, and the *second sociology* focuses on "human animals" - the individuals, their relationships, and actions.

uncontrollability, one needs trust [Stompzka, 1999] in that *one actor takes risk* to rely the outcome of transaction *on the performance of other actors.*

Before deciding to proceed with transaction, Internet-shopper must consider the risk of the transaction as well as his expectation that the online-shop will perform accordingly. The more his expectation is, the more chance that Internet-shopper will place trust. Conversely, the less the expectation is, the less chance that Internet-shopper will place trust.

The Trust Game

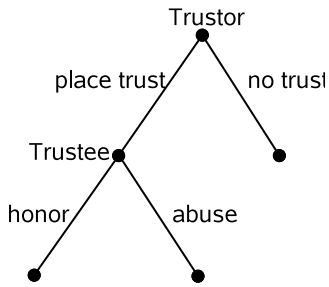


Figure 2.3: The Trust Game

Both Internet-shopper and online-shop play the trust game in which Internet-shopper plays as *Trustor,* who faces the trust dilemma, and the online-shop plays as *Trustee,* who is the subject of trust. Figure 2.3 illustrates the trust game where trustor has two choices; whether to place trust or not. When trustor does not place trust then the game end in which nobody receives or gains anything. On the other hand, when trustor decides to place trust, he opens the opportunity for the trustee whether to honor or to abuse the trust.

If online-shop decides to honor the trust by performing the delivery of goods and behaving as expected, then Internet-shopper will be happy and consequently the degree of expectation towards the online-shop increases. In contrast, if online-shop abuses the trust, then the Internet-shopper will be disappointed. This means that online-shop ruins the trust towards him and thus the degree of expectation decreases.

Evolution of Trust

McKnight and Chervany [Mcknight and Chervany, 1996] distinguish between *context independent trust* and *context dependent trust.* Context independent trust is about expectation that one has towards other. It is the degree of believe that someone will perform something that one expects, or in other words *expectation on trustee*

reliability. Context dependent trust is about one's action to rely to someone else; the decision to trust and to take risk.

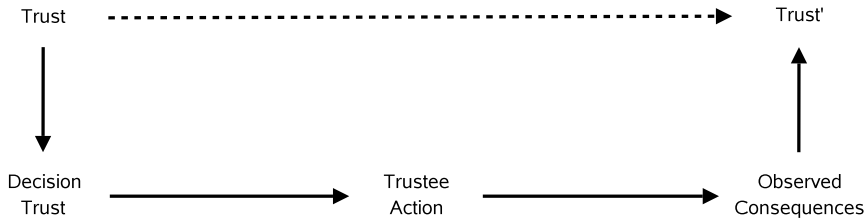


Figure 2.4: The Change of Context Independent Trust in a Trust Game

Figure 2.4 presents the evolution of context independent trust in a trust game. In the beginning of the game, trustor has certain degree of trust towards trustee which is used as considerations in the trust decision. Further, the actions of the trustee implies consequences observed by trustor. This observation alters the degree of context independent trust which will be used by trustor in the next game dealing with the particular trustee. When the game is repeated, one can see the dynamic of trust relation between a trustor and a trustee. After several transactions with particular online-shop, Internet-shopper gathers some experiences from which he can conclude the trustworthiness of online-shop.

Recommendation and Reputation System

In real life, it is seldom that one would take risk transacting with the stranger. How can trustor conclude trustworthiness of trustee when trustor has no previous experiences with the trustee. This is typical *zero knowledge* problem which one often finds in conducting transactions over Internet. In order to tackle this problem, human beings had developed mechanism of *recommendation* in which trustor gathers information from second hand referrals who have previous experiences with the particular trustee. These information are recommendations from others which supports the trustor in making decision trust.

The mechanism of recommendation makes use transitivity property of trust. A trustor trusts a trustee partly based on trust of other towards the trustee. Figure 2.5[3] shows the structures of trust relationship which some of them exhibits trust transitivity. The direct trust is direct trust relationship between a trustor and a trustee without any influence from other. The guarantee scheme introduces an intermediary who acts as guarantor for the trustee. The PKI, for example, establishes CA as the root of trust providing guarantee to other that any digital-certificate issued by the CA contain genuine public-key of the person mentioned in the certificate.

[3]Some of them are taken from Coleman [Coleman, 1990, p182]

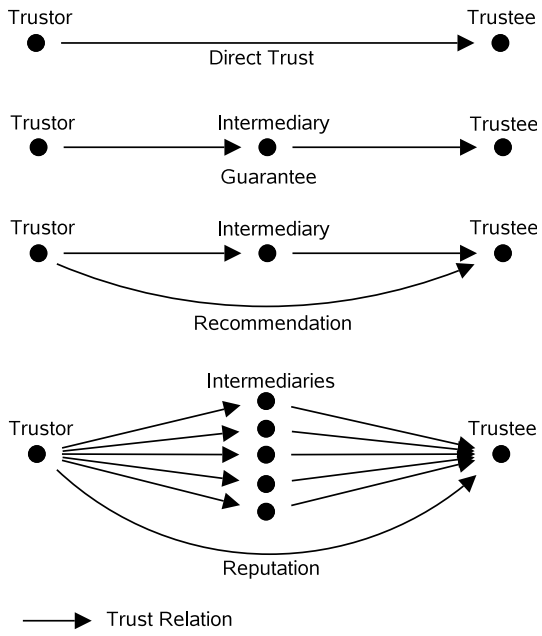


Figure 2.5: Structures of Trust Relationship

On the other hand, intermediary in recommendation scheme does not provide guarantee to the trustor. The intermediary only acts as second hand referral providing advisory to the trustor from which trust relationship from the trustor to the trustee can be initiated. The recommendations from all second hand referrals derives the reputation of trustee. It represents what is generally being said about the trustee.

Today's online-markets have been providing their users with the feedback and rating system which is based on the mechanism of recommendation and reputation. The system provides a centralized place where user can post their opinions about the results of previous experiences with other users for communal security. The biggest Internet auction site *Ebay*[4], for example, provides its users with the qualitative feedback and quantitative rating system in which those information are made public to be used by users in supporting their decision to trust particular user. Here, the whole ratings earned by each user are summed and computed by the system to derive reputation of the particular user and presented with the symbols and numbers. Instead of examining all qualitative feedbacks, a user is able to conclude trustworthiness of other by assessing correspondent quantitative value of reputation provided by system.

This system is well known as reputation system. It is based on quantitative method to express opinions or ratings, rating aggregation, as well as method to compute or derive reputation.

[4]http://www.ebay.com

2.4.4 Computational Trust

The effort to bring notion of trust in computer science came from Distributed Artificial Intelligent (DAI) area. Marsh [Marsh, 1994] is amongst the first who introduced this notion to computer science. He developed set of formalization of trust for DAI which is helpful to clarify the discussion of this notion and to develop common framework of trust. The formalization uses quantitative values of $[-1, +1)$ to express trust relation between two agents. It covers temporal considerations, i.e. how trust is changing through time, as well as non-temporal considerations, such as knowledge, importance, situational, and utility. The trusting decision is determined with cooperation threshold which takes into account several parameters including perceived risk and perceived competence as well as perceived degree of importance to cooperate.

Since then, the research have been heavily focused on method of trust computation for reputation systems in which trust is mostly perceived as *probability*. Josang et. al. [Jøsang et al., 2005] conducted survey on trust and reputation system for online service provision in which they identified six categories of reputation computation engines:

- Average System
  In this system, trustworthiness of trustee is expressed with numerical value and computed using simple summation and average. Ebay reputation system uses this simple model which is described by Resnick et al. [Resnick and Zeckhauser, 2002].

- Discrete Model
  In this model, trustworthiness of trustee is expressed using discrete value which represents qualitative measure. Abdul-Rahman and Hailes [Abdul-Rahman and Hailes, 2000], for example, use the range of discrete values from *Very Trustworthy*, *Trustworthy*, *Untrustworthy*, to *Very Untrustworthy*. In order to determine trustee reputation, trustor should use a lookup table in which the level of trustworthiness of trustee is upgraded / downgraded according to the rating from others.

- Bayesian Systems
  This system takes binary ratings as input and computes reputation using bayesian algorithm. The Beta reputation system [Ismail and Jøsang, 2002], for example, counts all positive experiences as well as negative experiences and uses Beta density functions to compute reputation scores.

- Belief Models
  Belief model was proposed by Josang [Josang, 1999] in which the belief of an

assertion, i.e. that agent x is trustworthy, is expressed with belief metrics (b, d, u) where b, d, and u represent belief, disbelief, and uncertainty, respectively. This model provides consensus operator to compute reputation. The operator takes the two paths of belief and combines them into one belief.

- Fuzzy Models
  This model incorporates fuzzy concept where trust and reputation is represented using membership functions which describe to what degree an agent can be described. Manchala [Manchala, 1998] proposed this scheme followed by Sabater and Sierra who proposed REGRET reputation system [Sabater and Sierra, 2001].

- Flow Models
  This system uses computation of trust and reputation based on transitive iterations through weighted looped arbitrarily chains. Google PageRank [Page et al., 1998] and Eigentrust [Sepandar D. Kamvar, 2003] reputation scheme for Peer-to-Peer network fall into this model.

Despite the rapid development of the method for computating trust and reputation, there are few, however, who study trust in the context of decision making as well as the the impact of reputation system in the system of agents. One of them is [Jøsang and Presti, 2004] who study decision trust in corelation with risk.

2.5 Summary

This chapter presented backgrounds and some works which are related with this thesis. First, structures of exchange in social system are reviewed in which some of important remarks are notified. Next, survey of mechanism to complete transactions are presented focusing on those that minimalizes the use of TTP. The last section discusses establishment of trust in computer mediated transactions. It gives the understanding of the research and the trend in using interpersonal trust to socially secure transactions.

Chapter 3

Transaction Basics

This chapter presents the development of system of accounts based on institutional facts as well as corresponding accounting mechanism. It begins by discussing and reformulating the problems in the respect of this research. Next, the concept of institutional fact is discussed to introduce the possibility to develop system of accounts. The next section presents transaction concept and its logic which is based on creation and elimination of institutional facts. Finally, a transaction algorithm is developed to show interactions and changes in the system in completing transactions.

3.1 Reformulating the Problem

Generally, people use money as medium of exchange in transactions. As mentioned in previous chapter, money is accepted as medium exchange because government legalizes its presence as well as guarantees the continuity of its use. The amount of money that a person possesses in his account represents the *state of finances* of the particular person.

During the transaction, money is physically being handed from one party to the transacting partner implying the change of the states of finances of transacting parties. The unique physical property of money circumvents the possibility for one to spend the same coin in two or more events of transactions. It establishes an accounting mechanism which is based on physical control.

The similar way applies also in electronic transactions. Transacting parties possess electronic accounts that are administered by the transaction server. The transaction server is under jurisdiction of trusted financial institution implying that the bits of information stored under one's account is regarded as the state of finances of the particular person.

When transaction occurs, the server manages alteration of those bits. Thus, the states of finances of transacting parties change. Here, overall structures of

the electronic transaction system establishes centralized accounting mechanism for transacting parties.

These illustrations show that in the system of transaction, *an actor possesses an account* which represents his state of finances. These accounts are present because of recognition of legal power of a trusted entity, e.g. the government in issuing the money, or the financial institution in administering electronic accounts. An event of transaction causes *the change of states of finances* of transacting parties that is regulated using mechanism of accounting.

With respect to this research, the questions that arise are twofold: (i) how to build the system of accounts representing states of finances of transacting parties in that the accounts are presence and recognized without legal power of single trusted entity, and (ii) how to develop accounting mechanism that regulates the change of those accounts in the event of transaction.

To answer those questions, the search was started in investigating various collaboration schemes which have been developed socially by human being.

A so-called *Arisan* [Miguel et al., 2005, Dharmawan, 2002], for example, is Indonesian informal savings club which exhibits an accounting mechanism based on social actions. In this club, all members gather once every period and collect an amount of money. A subset of members, who have their turn at the particular period, receive the money. A session of savings ends when every member have got the turn, and the new session starts with new set of random turns.

This social mechanism allows a member to "save" an amount of money to the club and collects them on the assigned period. The interesting part is, that members collectively witness who have received the money. Hence, everyone know who have not received the turn and thus is able to determine who are next to receive. Collective actions in witnessing the events of transaction emerges a collective accounting system. Here, part of the puzzle seems to have revealed, but how to socially present the accounts is yet to find.

3.2 Establishing the Accounts with Institutional Fact

The presence of accounts as part of system of transaction developed in this thesis is inspired by Searle's theory on the Construction of Social Reality [Searle, 1995, 2005b,a][1]. He explained how certain things emerge into reality as result of collective intentionality of whose to make use of them. Money, for example, is precious for those who share the intentionality that the piece of paper has economical value. Thus, it makes the money to be used as medium of exchange. On the other hand,

[1]The use of Searle's theory and the terminologies used in this chapter are arbitrary to be able to design the transaction algorithm. There have been lots of discussions, critics, as well as comments regarding the theory that can be found in [Smith and Searle, 2006, Smith, 2005].

the same piece of paper is worthless for those who don't share the same intentionality.

In order to state the logical form of his theory, Searle proposed the formula of statement:

"*X count as Y in context of C*"

which is known as count-as statement. To exemplify the formula, consider the following statement:

"*this piece of paper (X) counts as five euro-bill (Y) in Europe Union (C)*".

This statement is collectively accepted by citizens of all European countries that are united in Europe Union. They share the same intentionality that that piece of paper is money. Thus, it can be used as medium exchange within institution of Europe Union. Searle entitled this with *institutional fact*. The fact that exists relative to the context of institution and might not be the same in different context. Hence, that piece of paper is regarded as five euro-bill only within Europe Union and might not be used as money in Africa, for example.

The collective acceptance of a statement creating institutional fact enables the *deontic power*[2] of the fact. The fact that piece of paper in one's wallet is five euro bill enables one to pay dinner at restaurants in most of European countries. The restaurant ought to accept it not only because the owner shares the intentionality with his customer, but also that the socially constructed structures and the law on which the restaurant resides say that it is obligation for him to accept that piece of paper for the payment. The deontic power also enables the owner of the restaurant to pay his employees with that piece of paper, etc. Searle stated that what applies to money applies *mutatis mutandis* to marriage, property, government, and other institutional facts.

In the ontology of social reality, being piece of paper is *brute fact*. As contrary to Institutional fact, brute fact does not require the context of institution to occur. In the example above, money is the function which is collectively assigned to that piece of paper. Therefore, being emerged as money from a piece of paper is socially real. That is, the institutional fact emerges the reality of the object which is stated in the statement of the fact. Here, the collective acceptance of the fact gives those objects the deontic power to be regarded and to function to as stated in the fact. This inspiring work of Searle introduces a social scheme to answer the first question discussed earlier, namely to collectively emerge the accounts of transacting parties into reality.

To answer the second question consider the following illustration. There is a village of a folk living together without government and written law. In order to

[2] "*The powers that are constitutive of institutional facts are always matters of rights, duties, obligations, commitments, authorizations, requirements, permissions and privileges*" [Searle, 2005a]

establish property relation between a man and his house, the folk uses simple social recognition process. Collective acceptance of the statement "that house is property of Mr. White" creates the institutional fact that establishes property relation of that house to Mr. White. The deontic power gives Mr. White the right to use that house as his private residence or to rent it to other person. In order to sell the house to Mr. Black, White tells everyone in the folk that the house is sold to Mr. Black. Everybody knows and accepts in the first place that owner of that house is Mr. White. Therefore, the people allow this transaction to occur. That is, they collectively reject the statement they have been accepted which eliminates institutional fact establishing property relation of that house to Mr. White. At the same time, the folk collectively accept the new statement "that house is property of Mr. Black" creating institutional fact that establishes new relation property between that house and Mr. Black. Here, the new institutional fact overrides the prior right of ownership from Mr. White and transfer it to the new owner, Mr. Black.

This illustration shows that transaction can be carried out by collectively creating and eliminating institutional facts. This idea is the basis for the developing the transaction algorithm presented in this chapter.

3.3 Transaction Concept and Its Logic

Since the work of Searle, some authors [Jones and Sergot, 1996] [Artosi et al., 2004] have developed the formal logic of institutional fact which concentrated on the establishment institutional fact and its deontic power. This section, however, develops a simple logical notations of creation and elimination of institutional facts that helps to explain the transaction concept. It shall not be the attempt to investigate Searlean institutional-fact-hood, but rather a proposal for algorithms to accomplish the transaction.

The logical notations developed here is based largely on [Fagin et al., 1995] **S5** knowledge system which is also used by [Artosi et al., 2004] in formulating mutual belief in establishment of institutional fact. As companion to the explanation, figure 3.1 illustrates the creation and elimination of institutional fact among agents in a community.

3.3.1 The Building Blocks

Community of Agents

A community, denoted with C, is an institution that consists of n agents interested in conducting transaction with each other without the presence of trusted governor. An agent in community C is not only autonomic entity, but also social entity. It bases

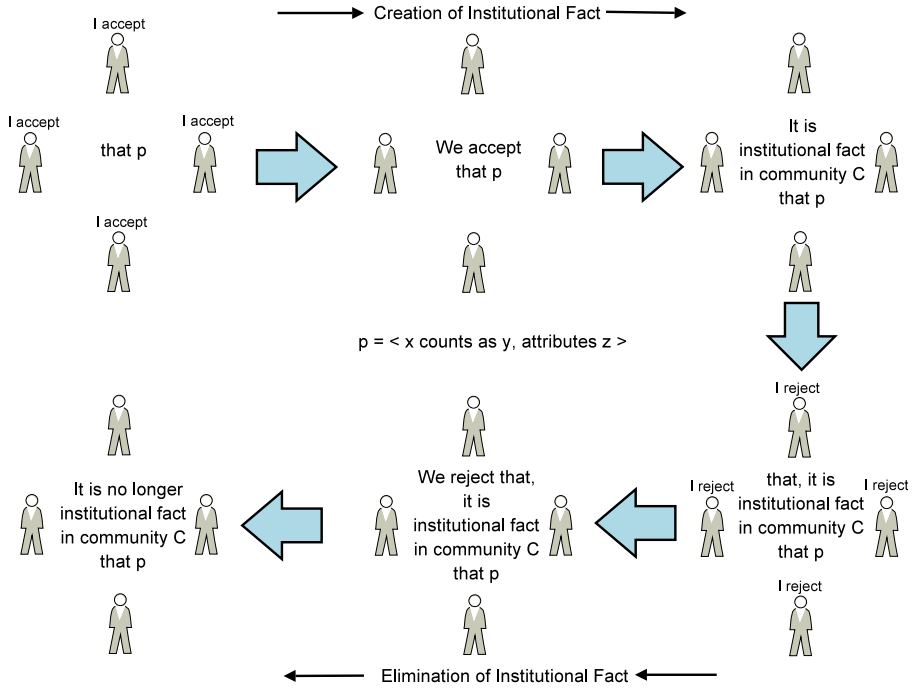


Figure 3.1: Creation and Elimination of Institutional Fact

its actions not only for the affair of itself but also for the affair of community, namely participating in collective actions to complete transactions for mutual benefit.

Arbitrary Assertions p

According to Searle, Institutional fact is based on count-as assertion that is collectively accepted by agents. The arbitrary assertion, denoted with f, is a triple $\langle x, y, z \rangle$ that is to be read "*this assertion has identification of x, y is arbitrary relation property as which x is counted to be, and z is set of pair-attributes of x*".

Fundamental Axioms of Creation of Institutional Fact

Creation of institutional fact is based on acceptance of arbitrary assertion p. Modal operator α is introduced to state "acceptance" of an assertion. Let 1 and 2 be two of agents in community C. Using modal operator of acceptance, $\alpha_1 p$ states that agent 1 accepts assertion p. Suppose agent 1 communicates his acceptance of p to agent 2 implying that agent 2 knows $\alpha_1 p$. Fagin et. al. denotes this with $K_a \alpha_1 p$ that reads "agent 2 knows that agent 1 accept p".

If agent 1 communicates $\alpha_1 p$ thorough agents, then everyone in community C knows that $\alpha_1 p$. This is denoted with the Fagin's modal operator E such that $E_C \alpha_1 p$ that reads "every agent in community C knows that agent 1 accepts p".

The uppercase C as subscript in E_C means every agent in C. $E_C\alpha_1 p$ is true exactly if everyone in community C knows $\alpha_1 p$:

$$E_C\alpha_1 p \iff \bigwedge_{a \in C} K_a\alpha_1 p$$

Consider that all agents in C accept p. They communicate their acceptance with each other such that every agent knows that every agent in community C accept p:

$$E_C\forall i \in C : \alpha_i p \iff \bigwedge_{a \in C} K_a\forall i \in C : \alpha_i p$$

It brings the community C to the state which is *considerably sufficient* for agents to realize that they have collectively accepted p:

$$E_C\forall i \in C : \alpha_i p \implies \alpha_C p \tag{3.1}$$

$\alpha_C p$ reads "community C collectively accept p", or "p is collectively accepted by community C". This is the fundamental basis of the creation of the institutional fact:

$$\alpha_C p \iff \odot_C p \tag{3.2}$$

where $\odot_C p$ is to be read *"it is institutional fact in community C that p"*. That is, an institutional fact is created if only if when all agents collectively accept the corresponding assertion, and this acceptance is known by every agent in the community.

Note that in this proposed transaction algorithm, $E_C\forall i \in C : \alpha_i p$ is considerably sufficient for agents to realize that collective acceptance has been reached. There is no need to achieve common knowledge, which by Fagin et. al. is denoted with C_C sufficiently achieved from E_C^k where $k = \{1, 2, ...\}$, since the unending loops to achieve this is expensive.

Fundamental Axioms of Elimination of Institutional Fact

Elimination of institutional fact is accomplished more or less in the same way that it was on its creation. Here, modal operator φ is introduced to state rejection of institutional fact, such that $\varphi_1 \odot_C p$ states that agent 1 rejects institutional fact established from assertion p.

Consider that $\odot_C p$ and agents in community want to eliminate $\odot_C p$ to complete a transaction, for example. To accomplish this, every agent rejects $\odot_C p$ and let others know its rejection:

$$E_C\forall i \in C : \varphi_i \odot_C p \iff \bigwedge_{a \in C} K_a\forall i \in C : \varphi_i \odot_C p$$

that brings the community C to the state which is *considerably sufficient* for agents to realize that they have collectively rejected $\odot_C p$:

$$E_C \forall i \in C : \varphi_i \odot_C p \implies \varphi_C \odot_C p \tag{3.3}$$

Here, $\varphi_C \odot_C p$ reads *"community C collectively reject that it is institutional fact in community C that p"*. Finally, basis for elimination of institutional fact is established:

$$\varphi_C \odot_C p \iff \otimes_C p \tag{3.4}$$

where $\otimes_C p$ is to be read *"it is no longer institutional fact in community C that p"*. To sum up, an institutional fact is eliminated if only if when all agents collectively reject it and this rejection is known by every agent in the community C.

Fundamental Axioms of Approval of Transaction Proposal π

As illustrated by the case of Mr. White and Mr. Black, to conduct a transaction, agents collectively accept set of assertions establishing new institutional facts, as well as collectively reject set of already established institutional facts. In order to accomplish this, transacting agents initiate transaction by communicating these information to other agents in a transaction proposal. The transaction proposal, denoted with π, is tuple $\langle P_n, P_o \rangle$ where P_n is set of proposed new assertions to be accepted, and P_o is set of institutional facts called to be rejected.

A modal operator λ is introduced to state approval of transaction proposal, such that $\lambda_1 \pi$ states that agent 1 approve π. If agent 1 communicates this approval to others then every agent knows that agent 1 approve π ($E_C \lambda_1 \pi$). If all agent approve π and communicate their approval with each other then all agents know that they have collectively approve π:

$$E_C \forall i \in C : \lambda_i \pi \implies \lambda_C \pi \tag{3.5}$$

In the ideal condition, collective approval of π by community C should represent the will of all agents in community. In certain condition or arrangement, however, collective approval of π by number of m agents, where $\frac{n}{2} < m < n$, may be sufficiently recognized as collective approval of π by the majority of community:

$$E_C \exists_{\geq m} i \in C : \lambda_i \pi \implies \lambda_C \pi \tag{3.6}$$

Here, $\exists_{\geq m} i$ is the quantifier "there exists at least m is such that". In this respect, condition represented in 3.6 superimposes condition in 3.5, such that the sum of all agents in the community (n) is always greater or equal then the sum of agents (m) which is arbitrary chosen to represent majority of community.

The state when collective approval of transaction proposal π is reached is the state when the corresponding transaction is allowed to occur. Thus, $\lambda_C\pi$ is defined to be the establishment of the transaction it self, i.e. that agents collectively accept all assertions in P_n and collective reject all institutional facts in P_o:

$$\lambda_C\pi \stackrel{def}{=} \forall p \in P_n : \alpha_C p \ \wedge \ \forall q \in P_o : \varphi_C q \tag{3.7}$$

That is, if all agents (or majority of agents) approved a transaction proposal and every agent in community know this approval, then it is assumed that transaction had established at the same time that the proposal was approved. Thus, at this state they have collectively accepted new assertions creating new institutional facts as well as collectively rejected institutional facts contained in the proposal.

3.3.2 Components of Transactions

There are two types of assertions establishing two types institutional facts used in the transactions. The first, is the institutional membership that is important to establish recognition of membership of an agent in the community. The second is Institutional-Money that establishes recognized medium exchange among agents in the community. To ease the notation, assertions from the first type will be denoted with $\hat{p}$ (hat) and the second with $\tilde{p}$ (tilde). The following passages describe those two in detail.

Institutional-Membership

Institutional-Membership is institutional fact establishing the membership of an agent in community C. It is based on assertion $\hat{p} = \langle id_{\hat{p}}, membership, \{(agentid, b), (financestate, v)\}\rangle$. $id_{\hat{p}}$ is unique identification of $\hat{p}$, *membership* is semantically agreed term of "being a member" of a group or community, and $(agentid, b)$ is the attribute denoting identifier of the agent, which in this example is b. The $(financestate, v)$ attribute denotes the value of financial status or state of finances of corresponding agent that is recognized by community. This attribute is used only in the transaction scheme based on alteration of state of finances which will be described later.

Collective acceptance of $\hat{p}$ ($\alpha_C\hat{p}$) creates an institutional-membership $\odot_C\hat{p}$ that is the fact establishing $id_{\hat{p}}$ as recognition of membership of agent a in community C. In order to ease the description, the corresponding agent is denoted using subscript. Thus, institutional-membership of agent $b \in C$ is denoted with $\odot_C\hat{p}_b$ and one of agent $s \in C$ with $\odot_C\hat{p}_s$.

Institutional-Money

Institutional-Money is institutional fact establishing community money intended to be used as general exchange medium in the trade. Institutional-Money, denoted with $\odot_C \tilde{p}$, is based on assertion $\tilde{p} = \langle id_{\tilde{p}}, money, \{(value, v), (owner, id_{\hat{p}})\}\rangle$. Here, $id_{\tilde{p}}$ is unique identifier of $\tilde{p}$, *money* is semantically agreed terms of being medium of exchange, and $(value, v)$ is the attribute establishing economical value of the money. The $(owner, id_{\hat{p}})$ attribute establishes the owner of the money who is identified by his institutional-membership $id_{\hat{p}}$.

Collective acceptance of $\tilde{p}$ by community C ($\alpha_C \tilde{p}$) creates $\odot_C \tilde{p}$ that is the fact establishing $id_{\tilde{p}}$ as community money as well as the ownership relation of $id_{\tilde{p}}$ to the owner-agent with corresponding institutional-membership is $id_{\hat{p}}$.

3.3.3 Transaction Schemes

In this system, transaction is defined as :

Definition 3.1: *Transaction is an exchange of trading objects between two or more agents causing changes in states of finances of transacting agents, that are established from institutional facts, all at once and consistently.*

The word "consistently" in above definition means that how much buyer pays should be as much as how much seller gets. Or in transaction conducted by three agents, how much the buyer pays should be the sum of how much the seller gets and how much the broker gets. Note that the system of transaction presented in this thesis assumes that any agreement between transacting agents is already established before they initiate the transaction.

This section presents (but not limited to) two schemes which can be carried out using the transaction system that base on creation and elimination of institutional facts. The first is based on the exchange of Institutional-Money and the second is based on the alteration of states of finances of transacting agents.

Transaction based on Exchange of Institutional-Money

This type of transaction needs two types of institutional fact: Institutional-Money and institutional-Membership. A transaction consists of the elimination of institutional-ownership establishing the ownership of the money to the buyer, and the creation of new institutional-ownership establishing the ownership of the money to the new owner the seller.

Consider that $b \in C$ (membership $id_{\hat{p}_b}$ is established from $\odot_C \hat{p}_b$) wants to buy goods from $s \in C$ (membership $id_{\hat{p}_s}$ is established from $\odot_C \hat{p}_s$) with the price of 100. Currently, b possesses money $id_{\tilde{p}}$ with value of 100 that is established from $\odot_C \tilde{p}$

based on assertion $\tilde{p} = \left\langle id_{\tilde{p}}, money, \left\{ \left(value, 100 \right), \left(owner, id_{\hat{p}_b} \right) \right\} \right\rangle$. Note that the owner of the money as stated in the fact is b.

Both transacting agents initiate transaction by proposing a transaction proposal π containing proposed new assertion $\tilde{p}' = \left\langle id_{\tilde{p}}, money, \left\{ \left(value, 100 \right), \left(owner, id_{\hat{p}_s} \right) \right\} \right\rangle$ that shall establish new ownership of the same money to agent s, as well as Institutional-Money called to be rejected $\odot_C \tilde{p}$, such that $\pi = \langle \{\tilde{p}'\}, \{\odot_C \tilde{p}\} \rangle$. In order to complete transaction, each agent approve π and communicate this approval to reach $\lambda_C \pi$. In reaching $\lambda_C \pi$, by definition every agent knows that they have accepted $\tilde{p}'$ establishing $\odot_C \tilde{p}'$ and rejected $\odot_C \tilde{p}$ establishing $\otimes_C \tilde{p}$. This can be formulated with:

$$\begin{aligned} \lambda_C \left\langle \{\tilde{p}'\}, \{\odot_C \tilde{p}\} \right\rangle \quad &\equiv \quad \alpha_C \tilde{p}' \wedge \varphi_C \odot_C \tilde{p} \\ &\Longrightarrow \quad \odot_C \tilde{p}' \wedge \otimes_C \tilde{p} \end{aligned}$$

After the transaction, community recognizes that $id_{\tilde{p}}$ is no longer property of b as result of collective rejection of $\odot_C \tilde{p}$ (which yields $\otimes_C \tilde{p}$), and that $id_{\tilde{p}}$ is property of s as result of $\odot_C \tilde{p}'$. That is, they have collectively transferred the money previously owned by buyer to the seller.

Transaction based on Altering States of Finances

Unlike its counterpart, this scheme needs only one type of institutional fact, namely institutional-membership. Here, the state of finances or financial status of an agent can be attached directly as an attribute in the corresponding institutional-membership. Transaction is completed by eliminating institutional-memberships of both transacting agents and creating new institutional-memberships containing altered states of finances.

Consider the same agents b and s who want to conduct transaction such that b should pay s as much as the value of val. Currently, institutional-membership of b is $\odot_C \hat{p}_b$ established from collective acceptance of assertion $\hat{p}_b = \langle id_{\hat{p}_b}, membership, \{(agentid, b), (financestate, v_b)\} \rangle$, and institutional-membership of s is $\odot_C \hat{p}_s$ established from collective acceptance of assertion $\hat{p}_s = \langle id_{\hat{p}_s}, membership, \{(agentid, s), (financestate, v_s)\} \rangle$.

Firstly, transacting agents b and s agree on the new assertions $\hat{p}'_b = \langle id_{\hat{p}_b}, membership, \{(agentid, b), (financestate, v'_b)\} \rangle$ and $\hat{p}'_s = \langle id_{\hat{p}_s}, membership, \{(agentid, s), (financestate, v'_s)\} \rangle$ which will be their new institutional-membership containing altered states of finances. Here, the value of states of finances in the new assertions are respectively $v'_b = v_b - val$ and $v'_s = v_s + val$.

Both agents initiate transaction by proposing a transaction proposal $\pi = \langle P_n, P_o \rangle$ such that $P_n = \{\hat{p}'_b, \hat{p}'_s\}$ and $P_o = \{\odot_C \hat{p}_b, \odot_C \hat{p}_s\}$. In order to accom-

plish transaction, each agent approves π and communicate his approval to others such that $\lambda_C \pi$ is reached. Upon reaching $\lambda_C \pi$ agents know that the corresponding transaction, proposed from π, occurs. Thus, agents have collectively "transfer" a value of *val* from agent b to agent s. Again, using logical notations previously described, this can be formulated with:

$$\begin{aligned} \lambda_C \left\langle \left\{ \hat{p}'_b, \hat{p}'_s \right\}, \left\{ \odot_C \hat{p}_b, \odot_C \hat{p}_s \right\} \right\rangle \quad &\equiv \quad \alpha \hat{p}'_b \wedge \alpha_C \hat{p}'_s \wedge \varphi_C \odot_C \hat{p}_b \wedge \varphi_C \odot_C \hat{p}_s \\ &\Longrightarrow \quad \odot_C \hat{p}'_b \wedge \odot_C \hat{p}'_s \wedge \otimes_C \hat{p}_b \wedge \otimes_C \hat{p}_s \end{aligned}$$

3.3.4 Remarks

Coleman's model of macro-micro-macro state transition in social system [Coleman, 1990] helps to understand how the change happened in macro level is the result of changes or combined actions in individual level. Figure 3.2 illustrates state transition of the system during a transaction:

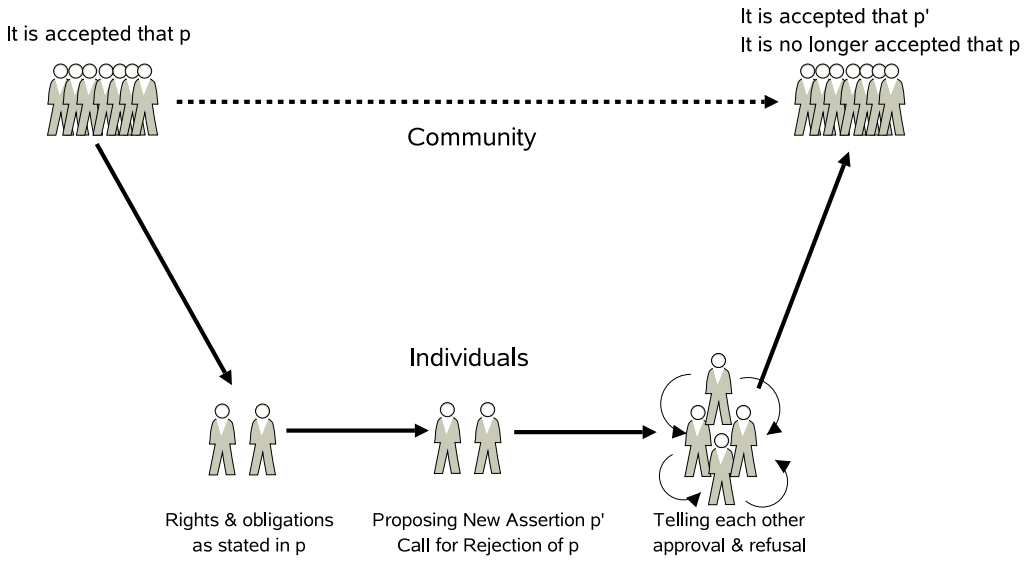


Figure 3.2: State transition using Macro-Micro-Macro model

- (macro→micro) institutional facts created in community level have deontic power that allow agents in individual level to use object or statement that is stated in the fact
- (micro→micro) exchanges or trading are allowed and happened among agents empowered by institutional facts

- (micro→macro) transaction is initiated by proposing transaction proposal to the community level, in which, agents approve transaction proposal implying collectively rejection of institutional facts and collectively acceptance of new statements establishing institutional facts

- thus, in macro level, the approval of transaction alters the synchronized knowledge of agents about the list of actual institutional facts

Proposed transaction scheme above shows that collective actions implying state transition of the system are done in decentralized manner such that the facilitation and mediation by trusted third party is not necessary. Thus, this transaction concept enables the presence of financial accounts as well as the system of accounting of those accounts for completing the transaction among decentralized agents.

3.4 Transaction Algorithm

Transaction concept and its logic presented in the previous section delineate more on the establishment of accounts of finances and accomplishment of transactions in logical form, but less in detailed actions of agents. This section takes one step further by presenting transaction algorithm applying the transaction concept.

The work in this section concentrates on specifying actions of agents from initiating transaction to the collective actions in completing initiated transaction, as well as showing corresponding changes in system variables caused by actions. The design goal is to have transaction algorithm which is based on the fundamental axioms of transactions described in the previous section.

Furthermore, this algorithm is intended to be generic for all possible schemes of transaction which is based on creation and elimination of institutional facts. Thus, it does not explicitely show the context of the transactions, e.g. the content of assertions or the relation of institutional facts with transacting agents, and most importantly it does not specify the reason for approval or refusal of transaction proposal in respect of specific transaction scheme. Nevertheless, there is function in the algorithm, entitled with *isPropValid*, is there to be the place for specific transaction schemes, e.g. to prevent double spending attempt in the transaction scheme based on exchange of institutional-money.

Transaction algorithm is specified in Lamport's Temporal Logic of Actions (TLA+)[Lamport, 2003, 2001][3]. Here, the use of TLA+ allows the specification to explicitely show the actions of agents as well as the change of system variables. A

[3]Brief knowledge of TLA+ is necessary to be able to understand this specification. Nevertheless the narrative description in this section attempts to minimize the need of mastering of TLA+ before one is able to read and understand provided specification. TLA+ related materials can be found at Lamport's site or at TLA+ page at RVS's site.

TLA+ specification assumes a state machine in which a state transition is occurred when a state predicate or an action is true given certain pre-condition. An Predicate Action Diagram (PAD) [Lamport, 1995], which is basically a state-machine, of corresponding specification is presented in figure 3.4 on page 42 to aid the explanation. As companion to the diagram, table 3.3 on page 43 and table 3.4 on page 44 provide detailed description of state transitions in the PAD showing the change of system variables. The complete specification can be found in the appendix A.

3.4.1 Agents and System Variables

Generally, a system consists of objects which engage and constitute system behavior. Behavior is sequence of states and a state is a snapshot of the value of system variables.

The system of communal governed transaction consists of agents who communicate through passing messages to engage in collaboration to complete transactions. Every agent has internal knowledge that is changed by agent's own action specified by algorithm, e.g. upon receiving certain message from other agent. Thus, internal knowledge of all agents constitute the system variable.

In the specification, an agent is represented with its internal knowledge. Denoted with *agent*, agent's internal knowledge is defined as several records which in TLA is specified as follows:

$$
\begin{array}{l}
\mathit{TypeAgent} \triangleq \\
\quad \land\ \mathit{agent} \in [\mathit{community} \\
\qquad\qquad \to [\mathit{ifacts} \quad\quad : \{f \in \mathit{p\_assertions} : \text{TRUE}\}, \\
\qquad\qquad\quad \mathit{msgfifo} \quad\ : \mathit{Seq}(\mathit{p\_messages}), \\
\qquad\qquad\quad \mathit{experience} : [\mathit{community} \\
\qquad\qquad\qquad\qquad\quad \to [\mathit{sum\_good} \ : \mathit{Nat}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_bad} \ : \mathit{Nat} \ ]], \\
\qquad\qquad\quad \mathit{proposals} \ : [\mathit{p\_proposals} \\
\qquad\qquad\qquad\qquad\quad \to [\mathit{proposed} \quad\quad : \text{BOOLEAN} , \\
\qquad\qquad\qquad\qquad\qquad \mathit{decided} \quad\quad : \text{BOOLEAN} , \\
\qquad\qquad\qquad\qquad\qquad \mathit{rated} \quad\quad\ : \text{BOOLEAN} , \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_approved} : \mathit{Nat}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_notapproved} : \mathit{Nat}]] \\
\qquad\qquad\quad] \\
\qquad\qquad]
\end{array}
$$

Additionally, the details of these records are explained in table 3.1.

Using this art of definition, it is easy to refer to a record of certain agent. For example, $\mathit{agent}[a].\mathit{proposals}[\pi].\mathit{decided}$ refers to the status of community decision

Record	Type	Description
ifacts	subset of *p_assertions*	Set of actual institutional facts known by agent. It is defined as subset of all possible assertions *p_assertions*. If community approve a transaction proposal containing assertions to be accepted P_n and institutional facts to be rejected P_o, every agent alters his *ifacts* such that P_o is subtracted from *ifacts* and P_n is added to *ifacts*. Thus, *ifacts* always contains actual institutional facts
msgfifo	sequence of *p_messages*	Buffer of sequence of messages received by an agents from communication channel. Agent checks the messages received in the buffer an act accordingly. If a message received has corresponding action then agent subtract the message from the buffer after the action is completed.
proposals	*p_proposal* → *proposed* : *Boolean* *decided* : *Boolean* *rated* : *Boolean* *sum_approved* : *Natural* *sum_notapproved* : *Natural*	Set of records containing statuses of all possible proposals known by agent. *proposed* indicates whether the proposal is currently being proposed. *decided* indicates whether community has decided to approve the particular proposal. *rated* indicates whether approved proposal (transaction) is already rated. *sum_approved* and *sum_notapproved*, respectively, count statements of approval and refusal during the completion of transaction.
experience	*community* → *sum_good* : *Natural* *sum_bad* : *Natural*	Set of reputation records of all agents in community based on rating / experiences resulted from past transactions with the particular agent. Since the transaction algorithm uses Beta reputation as example, it contains *sum_good* to count good experiences and *sum_bad* to count bad experiences with particular agent.

Table 3.1: Records of Internal knowledge of *agent*

toward proposal $\pi \in$ *p_proposals* which is known by agent $a \in$ *community*. Here, community is denoted deliberately with *community*, instead of C as it was in previous section, in order to provide clearness in the specification.

In this definition, institutional facts are contained in a record entitled with *ifacts*. Provided with transaction schemes presented in previous section, one can bear in

mind that agents memorize assertions known to be collectively accepted establishing institutional facts, and forget assertions of institutional facts that have been collectively rejected. In the algorithm, to memorize an assertion means to add the assertion into the *ifacts* and to forget the assertion means to subtract that assertion out of *ifacts*. Thus, *ifacts* is set of actual institutional facts known by agent.

3.4.2 Communication Channel

Communication channel is modeled as buffer entitled with *msgfifo*. It is a sequence containing messages that is transmitted among agents to accomplish the transaction. An agent sends a message by appending the message to the recipient's *msgfifo*. The *Broadcast* action presented below is an example of action in that set of agents $senders \subseteq community$ send message *msg* to any other agents in the *community*.

$$
\begin{aligned}
&Broadcast(senders,\ msg) \triangleq \\
&\quad \land\ senders \in \text{SUBSET } community \\
&\quad \land\ msg \quad\ \in p\_messages \\
&\quad \land\ agent' = [agent \text{ EXCEPT } ![\forall\ s \in community : s \notin senders] = \\
&\qquad\qquad\qquad [@ \text{ EXCEPT } !.msgfifo = \\
&\qquad\qquad\qquad\quad Append(@.msgfifo,\ msgStruct[senders,\ msg])]]
\end{aligned}
$$

The $agent'$ (variable - prime) is TLA way in denoting *state function* which assigns a value to a variable in the next state. As specified by *Broadcast* action, the value of the records in variable $agent'$ are assigned with the same value of those at the current state *except* the value of *msgfifo* of all agent $s \in community$ and $s \notin senders$ which are appended with *msg*.

3.4.3 Specifying Agent Actions in Completing Transaction

The specification begins with

$$Spec \triangleq InitAgent \land \Box[Next]_{agent}$$

where *Spec* denotes the specification of the system, *InitAgent* is the initial action to assign initial value of variable *agent*, and *Next* is the action done in every state step of the system. This specification means that *InitAgent* occurs only once at the first state of the system, and *Next* occurs in every state of the system behavior. Notation $\Box$ is temporal logic operator for "always" such that $\Box Next$ asserts that *Next* is always true in every step of system behavior. $\Box\,[Next]_{agent}$ means that every step of *Next*, which is always true, either changes variable *agent* to its next value or leaves it unchanged. This is the way of TLA to express continues system behavior.

Next is defined as follows:

$$
\begin{aligned}
&\mathit{Next} \triangleq \\
&\quad \lor \mathit{Encounter} \\
&\quad \lor \mathit{ReceiveProposalApprove} \\
&\quad \lor \mathit{ReceiveProposalRefuse} \\
&\quad \lor \mathit{ReceiveStatementApproval} \\
&\quad \lor \mathit{ReceiveStatementRefusal} \\
&\quad \lor \mathit{ReceiveStatementUndetermined} \\
&\quad \lor \mathit{ProcCommunityApproval} \\
&\quad \lor \mathit{ProcCommunityRefusal} \\
&\quad \lor \mathit{ProcRatingGood} \\
&\quad \lor \mathit{ProcRatingBad} \\
&\quad \lor \mathit{ReceiveRatingGood} \\
&\quad \lor \mathit{ReceiveRatingBad} \\
&\quad \lor \mathit{ReceiveRatingUndetermined}
\end{aligned}
$$

Since *Next* is always true, every action stated in the definition of *Next* is true or enabled in every state of the system behavior if only if the condition specified in the corresponding action is fulfilled. The exception is for *Encounter* in which no pre condition is specified.

The concept of encounter, denoted by *Encounter*, is introduced to establish the event where two agents *buyer* and *seller* meet and propose a transaction proposals to the community. It is the initiator of transaction that shall be the result of join intention and agreement between the two agents[4]. *Encounter* is specified as:

$$
\begin{aligned}
&\mathit{Encounter} \triangleq \\
&\quad \land \langle \mathit{buyer}, \mathit{seller} \rangle \\
&\qquad\qquad [\text{CHOOSE } \langle x, y \rangle \in \mathit{community}^2 : x \neq y] \\
&\quad \land \langle P_o, P_n \rangle \\
&\qquad\qquad [\text{CHOOSE } \langle O, N \rangle \in (\text{SUBSET } p\_\mathit{assertions})^2 : \\
&\qquad\qquad\qquad\qquad \land O \subseteq \mathit{agent}[\mathit{buyer}].\mathit{ifacts} \\
&\qquad\qquad\qquad\qquad \land \forall x \in O : x \notin N \\
&\qquad\qquad\qquad\qquad \land \forall x \in N : x \notin O \ ] \\
&\quad \land \mathit{lgratio}\ [\text{CHOOSE } x \in \mathit{Real} : x > 0] \\
&\quad \land \mathit{Propose}(\mathit{buyer}, \mathit{seller}, P_o, P_n, \mathit{lgratio})
\end{aligned}
$$

[4]Since that this thesis is not about intelligent agents, algorithm presented here does not concern about self intention or join intention in transacting or trading.

which says: encountered *buyer* and *seller*, two agents selected from *community*, propose set of assertions P_n to be collectively accepted by community as well as call for rejection of another set of assertions P_o which at that state are accepted as institutional facts by the community. Parameter *lgratio* is not discussed here as it is subject for the next chapter.

As previously mentioned, TLA specification assumes state machine. Figure 3.4 shows a state machine that illustrates how transaction is completed after it was

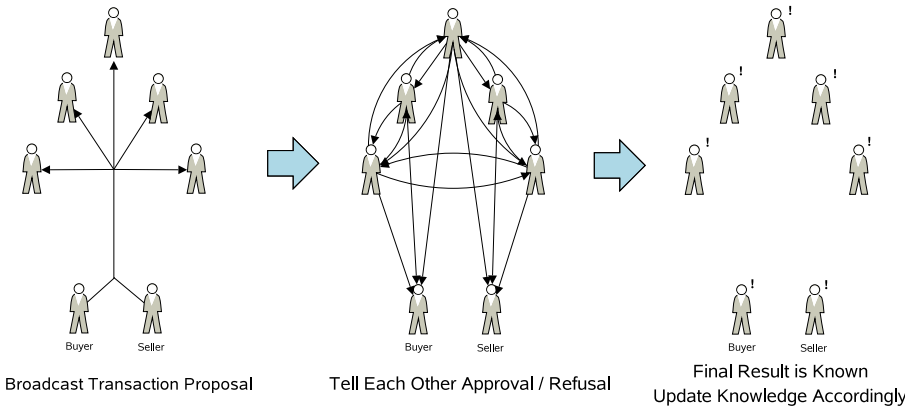


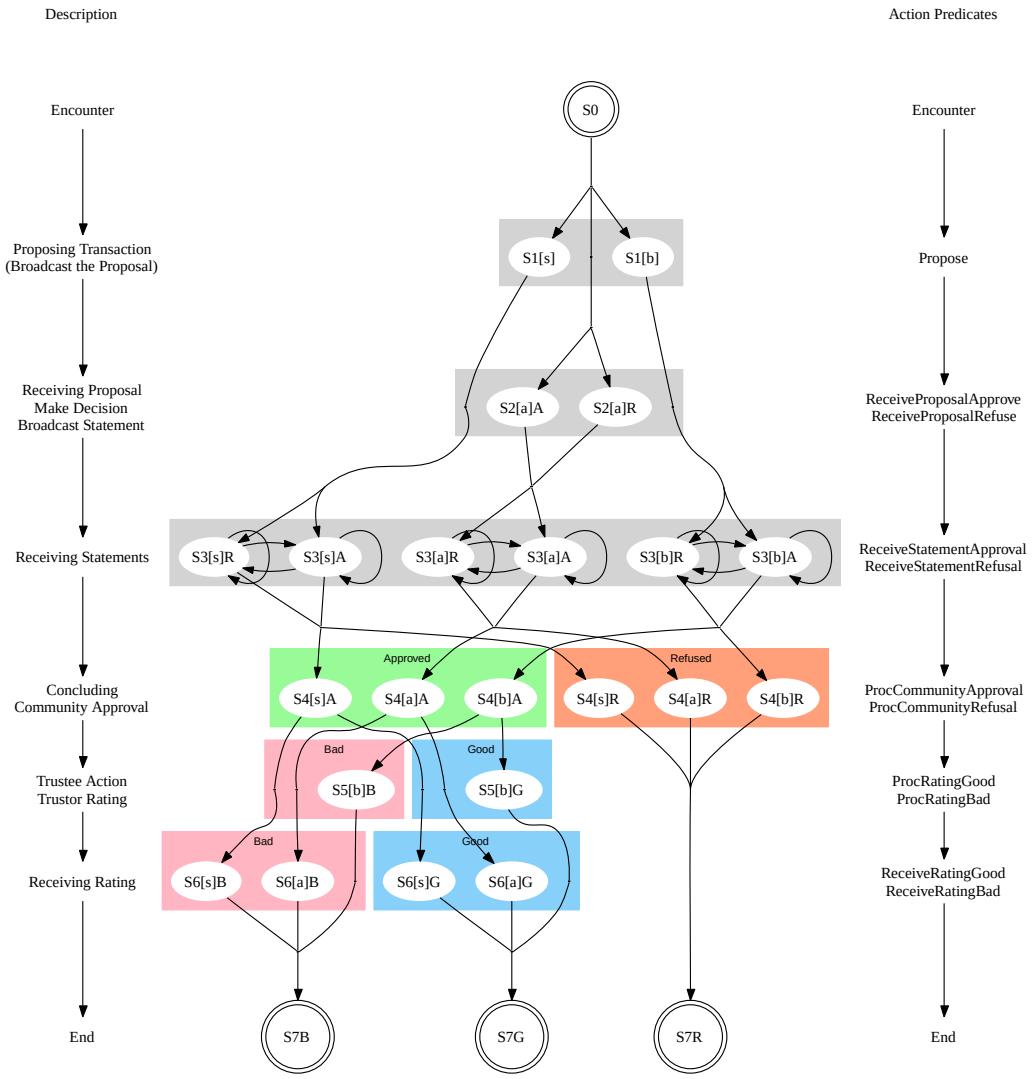
Figure 3.3: Illustration of Transaction Algorithm

initiated by *Encounter* at state S_0. Additionally, figure 3.3 illustrates agent actions according to transaction algorithm.

The state machine contains divergent arrows structures, e.g. $S_0 \rightarrow \{S_1[s], S_1[b], S_2[a]R, S_2[a]A\}$ and convergent arrows structures, e.g. $\{S_5[s], S_5[b], S_5[a]\} \rightarrow S_6A$ to show system state transition that consists of changes in agent level, and states of set of agents that constitute the system state, respectively. A state named with "[]" bracket represents the state of agent who is stated in the bracket, e.g. $S_5[s]$ is state of agent s. Conversely, the state entitled without the bracket represents the state where all agents have synchronized internal knowledge. The diagram encompasses the states of agent b, s, and a ($a \in Community, a \notin \{buyer, seller\}$) which respectively represent *buyer*, *seller*, and other agent in the community who collaborate to accomplish transaction.

To simplify the explanation and to be more focus on the completion of the transaction itself, nodes in the diagram refers only to the states of internal knowledge $agent[i].proposals$, $agent[i].ifacts$, and $agent[i].experience$, but not the communication channel $agent[i].msgfifo$. Therefore, some of the actions, such as, *ReceiveStatementUndetermined* and *ReceiveRatingUndetermined* are not discussed and self explanatory.

Furthermore, actions *ProcRatingGood*, *ProcRatingBad*, *ReceiveRatingRatingGood*, *ReceiveRatingBad*, *ProcRatingGood*, *ProcRatingBad*,



Node entitled with :
[b] indicates state of agent *buyer*
[s] indicates state of agent *seller*
[a] indicates state of any other agents

Figure 3.4: Predicate Action Diagram of Transaction Algorithm

State Transition	Pre Condition	**Action** / Description	Post Condition
$S_0 \rightarrow S_1[b]$ $S_0 \rightarrow S_1[s]$ $b = buyer$ $s = seler$	agent[{b,s}].proposals[π]. proposed=FALSE decided=FALSE rated=FALSE sum_approved=0 sum_notapproved=0 agent[{b,s}].ifacts	**Propose** *Encountered buyer and seller propose transaction proposal π to the rest of community, they mark their knowledge of p.proposed as true and counts their acceptance (2) in sum_approved.*	agent[{b,s}].proposals[π] proposed=TRUE <decided> <rated> sum_approved=2 <sum_notapproved> agent[{b,s}].<ifacts>
$S_0 \rightarrow S_2[a]A$ $a \in Community$ $a \notin \{buyer, seller\}$	agent[a].proposals[π]. proposed=FALSE decided=FALSE rated=FALSE sum_approved=0 sum_notapproved=0 agent[a].ifacts	**ReceiveProposalApprove** *Agent receives transaction proposal π from buyer and seller, and decides to approve. He adds 3 in sum_approved (buyer, seller, and himself), mark proposed as true, and broadcasts his decision*	agent[a].proposals[π] proposed=TRUE <decided> <rated> sum_approved+=3 <sum_notapproved> agent[a].<ifacts>
$S_0 \rightarrow S_2[a]R$ $a \in Community$ $a \notin \{buyer, seller\}$	agent[a].proposals[π]. proposed=FALSE decided=FALSE rated=FALSE sum_approved=0 sum_notapproved=0 agent[a].ifacts	**ReceiveProposalRefuse** *Agent receives transaction proposal π from buyer and seller, and decides to refuse it. He adds 2 in sum_approved (buyer, seller) and 1 in sum_notapproved (himself), marks proposed as true, and broadcasts his decision*	agent[a].proposals[π] proposed=TRUE <decided> <rated> sum_approved+=2 sum_notapproved+=1 agent[a].<ifacts>

The < > bracket indicates that the value of variable in the bracket does not change in post condition
The += has the same semantic as increment by operator += used in C language

Table 3.3: State Transitions (Part 1)

and function *trustDecision*, which correspond to S_5s, S_6s, and S_7s, are not discussed here as they are subjects of and defined in the next chapter. Thus, explanation in this chapter assumes function *trustDecision* always returns *true*.

3.4.4 Steps to Complete the Transaction

The following discussions explain steps to complete transaction by means of state transitions and corresponding actions aided by state machine in figure 3.4. A summary of variable changes by each action are presented in table 3.3 and 3.4. These descriptions do not show the algorithm in TLA specification explicitely, but rather series of state transitions. Complete specification is presented in appendix A.

This chapter discusses steps up to S_4. Since S_5, S_6 are the subject of the next chapter.

S_o Initial State

This is the initial state of a transaction where *buyer* and *seller* meet through the trigger by *Encounter*. They select institutional facts that should be collectively rejected P_o and new assertions that shall be collectively accepted

State Transition	Pre Condition	**Action** / Description	Post Condition
$S_1[s] \rightarrow S_3[s]A$ $S_1[b] \rightarrow S_3[b]A$ $S_2[a]R \rightarrow S_3[a]A$ $S_2[a]A \rightarrow S_3[a]A$ $a, b, s \in Community$ $a \notin \{b, s\}$	agent[{a,b,s}].proposals[π]. proposed=TRUE decided=FALSE rated=FALSE (sum_approved + sum_notapproved)<|agent| agent[{a,b,s}].ifacts	**ReceiveStatementApproval** *Agent receives a statement of approval from other agent and marks this by adding 1 in sum_approved*	agent[{a,b,s}].proposals[π]. <proposed> <decided> <rated> sum_approved+=1 <sum_ notapproved> agent[{a,b,s}].<ifacts>
$S_1[s] \rightarrow S_3[s]R$ $S_1[b] \rightarrow S_3[b]R$ $S_2[a]R \rightarrow S_3[a]R$ $S_2[a]A \rightarrow S_3[a]R$ $a, b, s \in Community$ $a \notin \{b, s\}$	agent[{a,b,s}].proposals[π]. proposed=TRUE decided=FALSE rated=FALSE (sum_approved + sum_notapproved)<|agent| agent[{a,b,s}].ifacts	**ReceiveStatementRefusal** *Agent receives a statement of refusal from other agent and marks this by adding 1 in sum_notapproved*	agent[{a,b,s}].proposals[π]. <proposed> <decided> <rated> <sum_approved> sum_ notapproved+=1 agent[{a,b,s}].<ifacts>
$S_3[s]A \rightarrow S_4[s]A$ $S_3[s]R \rightarrow S_4[s]A$ $S_3[b]A \rightarrow S_4[b]A$ $S_3[b]R \rightarrow S_4[b]A$ $S_3[a]A \rightarrow S_4[a]A$ $S_3[a]R \rightarrow S_4[a]A$ $a, b, s \in Community$ $a \notin \{b, s\}$	agent[{a,b,s}].proposals[π]. proposed=TRUE decided=FALSE rated=FALSE (sum_approved + sum_notapproved)=|agent| sum_approved$\geq$ (|agent| $\times$ mt) agent[{a,b,s}].ifacts	**ProcCommunityApproval** *Agent finds a condition where statements counters indicate that all agents have stated their statement and there are sufficient approvals that determines community approval on the proposal π. Agent updates his knowledge on currently accepted institutional fact according to record P_o and P_n stated in proposal π.*	agent[{a,b,s}].proposals[π]. proposed=FALSE decided=TRUE <rated> sum_approved=0 sum_notapproved=0 agent[{a,b,s}].ifacts= agent[{a,b,s}].ifacts $\setminus \pi.P_o \cup \pi.P_n$
$S_3[s]A \rightarrow S_4[s]R$ $S_3[s]R \rightarrow S_4[s]R$ $S_3[b]A \rightarrow S_4[b]R$ $S_3[b]R \rightarrow S_4[b]R$ $S_3[a]A \rightarrow S_4[a]R$ $S_3[a]R \rightarrow S_4[a]R$ $a, b, s \in Community$ $a \notin \{b, s\}$	agent[{a,b,s}].proposals[π]. proposed=TRUE decided=FALSE rated=FALSE (sum_approved + sum_notapproved)=|agent| sum_approved< (|agent| $\times$ mt) agent[{a,b,s}].ifacts	**ProcCommunityRefusal** *Agent finds a condition where statements counters indicate that all agents have stated their statement but there are insufficient approvals that determines community refusal on the proposal π. It resets the state of proposal p on his knowledge such that the records of p are the same on those at state S_0*	agent[{a,b,s}].proposals[π]. proposed=FALSE <decided> <rated> sum_approved=0 sum_notapproved=0 agent[{a,b,s}].<ifacts>

The < > bracket indicates that the value of variable in the bracket does not change in post condition
The += operator has the same semantic as "increment by" operator += used in C language

Table 3.4: State Transitions (Part II)

P_n, to be included in a transaction proposal π. At this stage, communication buffer *msgfifo* of every agent is assumed to be empty and records of the status of transaction proposals in the knowledge of every agent are

set to default (*proposed* = *FALSE*, *decided* = *FALSE*, *rated* = *FALSE*, *sum_approved* = 0, and *sum_notapproved* = 0). Furthermore, agents' knowledge of actual list of institutional facts *ifacts* are synchronized such that $agent[a].ifacts = agent[b].ifacts$ for all agent a and b that are members of the community.

Remark 3.1: *Given this pre-conditions, there is no other action stated in the Next that is enabled or true in this particular state, except Encounter. As previously discussed, Encounter is always true in every step of system behavior because* $\Box$*Next asserts Next to be always true and Encounter is defined to be enabled without full filling any pre-condition. Thus, the transaction is initiated only from Encounter.*

$S_0 \rightarrow \{S_1[s], S_1[b]\}$ **Propose**

After the encounter, *buyer* and *seller* initiate action *Propose*. Message *msg* containing the proposal π is broadcasted to the rest of community by appending *msg* in buffer *msgfifo* of each agent. Both transacting agents change the status of proposal π in their knowledge: *proposed* = *TRUE* indicating that the proposal π is currently being proposed and *sum_approved* = 2 indicating two agents who already approved the proposal, namely *buyer* and *seller*.

During the next step, each agent a, where $a \in Community$ and $a \notin \{buyer, seller\}$ finds message containing proposal π in the *msgfifo*. Agent a verifies whether π is valid using function *isPropValid*, and makes trust decision according using function *trustDecision* which in this chapter is assumed to be *true*.

$S_0 \rightarrow S_2[a]A$ **ReceiveProposalApprove**

Action *ReceiveProposalApprove*, implying this particular state transition, is enabled if agent $a \in Community, a \notin \{buyer, seller\}$ finds message containing proposal π in the first sequence of *msgfifo*, and both *isPropValid* and *trustDecision* are *TRUE*. Agent a alters his knowledge on proposal π such that *proposed* = *TRUE* indicating that the proposal is currently being proposed, and *sum_approved* is increased by 3 because as far as agent a concerns, there are 3 agents (*buyer*,*seller*, and a himself) who approve the proposal in this state. Agent a broadcast the statement of approval to the rest of community including *buyer* and *seller* and deletes the message he just received from *msgfifo*.

$S_0 \rightarrow S_2[a]R$ **ReceiveProposalRefuse**

Action *ReceiveProposalApprove*, causing this particular state transition, is enabled if agent $a \in Community, a \notin \{buyer, seller\}$ finds message containing proposal π in the first sequence of *msgfifo* and either *isPropValid* and *trustDecision* is *FALSE*. Agent a alters his knowledge on proposal π such

that *proposed* = *TRUE* indicating that proposal is currently being proposed, *sum_approved* is increased by 2 indicating the two approvals from proposing agents (*buyer*,*seller*), and *sum_notapproved* is increased by 1 indicating *a*'s refusal. Agent *a* broadcast his refusal statement to the rest of community including *buyer* and *seller* and subtracts the message he just received out of *msgfifo*.

Remark 3.2: *Suppose one agent receive a proposal. From the definition of ReceiveProposalApprove, one might notice that both isPropValid and trustDecision must be TRUE in order this action to be enabled. Conversely, should one of isPropValid and trustDecision is FALSE or both of them are FALSE, then ReceiveProposalRefuse is enabled. Since* $\neg(isPropValid \wedge trustDecision) = \neg isPropValid \vee \neg trustDecision$*, in this particular step, ReceiveProposalApprove or ReceiveProposalRefuse are mutual exclusive for an agent, such that one agent can only engage once either ReceiveProposalApprove or ReceiveProposalRefuse. If i be the sum of agents who act ReceiveProposalApprove and j be the sum of agents who act ReceiveProposalRefuse, then total agents in community n is* $i + j + 2$*, where* 2 *represents buyer and seller.*

Upon receiving approval or refusal statement from others, agent pops the message from the buffer *msgfifo* one by one and counts the statement in the statement counters. The state machine presented figure 3.4 shows that there are circular arrow structures representing loops in receiving the statements.

$\{S_2[a]A \mid S_2[a]R \mid S_3[a]A \mid S_3[a]R\} \rightarrow S_3[a]A$ **ReceiveStatementApproval** ($\bigwedge$)
Action *ReceiveStatementApproval*, which causes this particular state transition, is enabled if an agent $a \in Community, a \notin \{buyer, seller\}$ finds message in the first sequence *msgfifo* containing statement of approval (*status* = *accept*) of proposal $\pi \in p\_proposals$, and status *proposed* of proposal π is *TRUE* (that π is being proposed at $S_2[a]A$ or $S_2[a]R$). Agent increment *sum_approved* by one which indicates that there is one acceptance received, and deletes the corresponding message from the buffer.

$\{S_1[s] \mid S_3[s]A \mid S_3[s]R\} \rightarrow S_3[s]A$ **ReceiveStatementApproval**
The same conditions and actions apply for agent *seller* as described in $\bigwedge$.

$\{S_1[b] \mid S_3[b]A \mid S_3[b]R\} \rightarrow S_3[b]A$ **ReceiveStatementApproval**
The same conditions and actions apply for agent *buyer* as described in $\bigwedge$.

$\{S_2[a]A \mid S_2[a]R \mid S_3[a]A \mid S_3[a]R\} \rightarrow S_3[a]R$ **ReceiveStatementRefusal** ($\bigvee$)
Action *ReceiveStatementRefusal*, which implies this particular state transition, is enabled if an agent $a \in Community, a \notin \{buyer, seller\}$ finds message in

the first sequence *msgfifo* containing statement of refusal (*status* $\neq$ *approve*) of proposal $\pi \in p\_proposals$, and status *proposed* of proposal π is *TRUE* (that π is being proposed at $S_2[a]A$ or $S_2[a]R$). Agent increment *sum_notapproved* by one which indicates that there is one received, and deletes the corresponding message from the buffer.

$\{S_1[s] \mid S_3[s]A \mid S_3[s]R\} \rightarrow S_3[s]R$ **ReceiveStatementRefusal**
The same conditions and actions apply for agent s as described in $\bigtriangledown$.

$\{S_1[b] \mid S_3[b]A \mid S_3[b]R\} \rightarrow S_3[b]R$ **ReceiveStatementRefusal**
The same conditions and actions apply for agent b as described in $\bigtriangledown$.

After receiving all statements toward proposal π, an agent can determine whether π is collectively approved or refused by the community. As stated by axioms 3.5 and 3.6, if the sum of approval (*sum_approved*) is equal or greater than number of majority of agents, of which the threshold is determined by constant m, then community has approved the transaction from proposal π, and thus transaction occurs. Conversely, community refuses the proposal, and hence transaction does not occur.

Remark 3.3: *In this particular stage, every agent should know the synchronized values of sum_approved and of sum_notapproved. To proof this, let n be total agents in community, i be the sum of agents who broadcast approvals (ReceiveProposalApprove), and j be the sum of agents who broadcast refusals (ReceiveProposalRefuse), such that $i + j + 2 = n$ (2 is for buyer and seller who already state their approval by proposing the transaction).*

From definition of Propose, both buyer and seller in $S_1[b]$ and $S_1[s]$, have sum_approved $= 2$ and sum_notapproved $= 0$. After receiving all statements from $i + j$ agents, both buyer and seller have sum_approved $= 2 + i$ and sum_notapproved $= j$. Thus, sum_approved + sum_notapproved $= 2 + i + j =$ total_agents.

From pre-condition in S_0 and definition of ReceiveProposalApprove, at the $S_2[a]A$ the i agents have sum_approved $= 3$ and sum_notapproved $= 2$. After receiving all approval statements from $i-1$ agents (1 represents him self) and j refusal statements from j agents, each agent in this set has sum_approved $= 3 + (i - 1) = 2 + i$ and sum_notapproved $= j$.

From pre-condition in S_0 and definition of ReceiveProposalRefuse, at the $S_2[a]R$ the j agents have sum_approved $= 2$ and sum_notapproved $= 1$. After receiving all approval statements from i agents and refusal statements from $j - 1$ agents (1 represents him self), each agent in this set has sum_approved $= 2 + i$ and sum_notapproved $= 1 + (j - 1) = j$.

Thus, at the end of this stage where all agents have received all statements from $i + j$ agents, they have the same values of sum_approved and sum_notapproved, where sum_approved + sum_notapproved $= 2 + i + j = n$.

$\{S_3[a]R \mid S_3[a]A\} \rightarrow S_4[a]A$ **ProcCommunityApproval** (†)

This action is enabled if agent $a \in Community, a \notin \{buyer, seller\}$ finds in his internal knowledge that all agents have given their statements towards proposal $\pi \in p\_proposals$ (indicated with $sum\_approved + sum\_notapproved =$ total agents in the community), and there are sufficient approval for π from the majority of community (indicated with *sum_approved* is greater or equal than [total agents $\times$ mt], where $0.5 \leq mt \leq 1$ - mt is majority threshold). Agent alters his knowledge about the list of actual institutional fact such that $ifacts' = ifacts \setminus \pi.P_o \cup \pi.P_n$ as well as statuses of π to the following values:$decided = TRUE$, $sum\_approved = 0$, $sum\_notapproved = 0$, $proposed = FALSE$.

$\{S_3[s]R \mid S_3[s]A\} \rightarrow S_4[s]A$ **ProcCommunityApproval**

The same conditions and actions apply for agent s as described in †.

$\{S_3[b]R \mid S_3[b]A\} \rightarrow S_4[b]A$ **ProcCommunityApproval**

The same conditions and actions apply for agent b as described in †.

$\{S_3[a]R \mid S_3[a]A\} \rightarrow S_4[a]R$ **ProcCommunityRefusal** (‡)

This action is enabled if agent $a \in Community, a \notin \{buyer, seller\}$ finds in his internal knowledge that all agents have given their statements towards proposal $\pi \in p\_proposals$ (indicated with $sum\_approved + sum\_notapproved =$ total agents in the community), and there are insufficient approval for π from the majority of community (indicated with *sum_approved* is less than [total agents $\times$ mt], where $0.5 \leq mt \leq 1$ - mt is majority threshold). Agent alters his knowledge about statuses of π to the default values as follows: $proposed = FALSE$,$decided = FALSE$, $rated = FALSE$, $sum\_approved = 0$, $sum\_notapproved = 0$. Since proposal is refused, there are no change to the list of actual institutional facts. Thus *ifacts* stays no change.

$\{S_3[s]R \mid S_3[s]A\} \rightarrow S_4[s]R$ **ProcCommunityRefusal**

The same conditions and actions apply for agent s as described in ‡.

$\{S_3[b]R \mid S_3[b]A\} \rightarrow S_4[b]R$ **ProcCommunityRefusal**

The same conditions and actions apply for agent b as described in ‡.

Here, community approval of transaction means that community collectively accept each assertion in proposed P_n and collectively reject each institutional fact stated

in P_o. This implies the alteration of actual list of institutional facts *ifacts* of each agent.

Conversely, no action is done to alter *ifacts* and thus *ifacts* of each agent does not change. Moreover, the statuses of π is roll-backed to the initial value by *ProcCommunityRefusal*. Thus, presumably that only one transaction proposal is processed at a time, the values of system variables at S_6R is equal to those at S_0.

Remark 3.4: *In this last stage of the completion of transaction, every agent should have the same conclusion. Let n be total agents in community and mt be the threshold of majority. The definition of ProcCommunityApproval state that this particular action is enabled when sum_approved* $\geq (n \times mt)$. *Conversely, the definition of ProcCommunityRefusal states is enabled when sum_approved* $< (n \times mt)$. *Thus ProcCommunityApproval and ProcCommunityRefusal are mutual exclusive for an agent in this stage. Since both actions are mutual exclusive for an agent, and that all agents have synchronized value of sum_approved and sum_notapproved in the pre condition, then obviously all agents reach at the same conclusion: whether proposal* π *is collectively approved or not, and act accordingly.*

3.4.5 Discussion

As stated in the beginning of the section, the transaction algorithm should fulfill fundamental axioms presented previous section. Let's review some remarks that were previously discussed:

- Remark 3.1 discussed that at the beginning all agents have synchronized knowledge, and that the transaction is initiated only through *Encounter*. By invoking *Propose*, transacting agents broadcast transaction proposal to other agents, say the collaborators. Here, it is assumed that by proposing transaction proposal, the transacting agents have already approved the proposal.

- Remark 3.2 discussed that each collaborator can only either approve or refuse the transaction proposal but not both. Each count the approval and the refusal and broadcast the statements to the rest of community including transacting agents.

- Remark 3.3 discussed all agents have synchronized knowledge of the approval and refusal statuses of proposed transaction after receiving all statements in this stage.

- Remark 3.4 showed that since all agents have synchronized knowledge of the approval and refusal statuses of proposed transaction, and since *ProcCommunityApproval* and *ProcCommunityRefusal* are mutual exclusive for

an agent, all agents reached at the same conclusion, the determination whether proposal is collectively approved or refused, but not both.

Consider following statements to state that the algorithm fulfill fundamental axioms:

- Since π is broadcasted among agents then each agent knows the P_o and the P_n contained in π.
- Agent knowledge of actual institutional fact is modeled as a set *ifacts* which is subset of all possible assertions *p_assertions*
- The previously reviewed remarks showed that algorithm leads the agents to know each other statements of approval/refusal of transaction proposal π, such that $E_C \forall_{\geq m} i \in C : \lambda_i \pi \iff \forall a \in C : \forall_{\geq m} i \in C : \lambda_i \pi$ is fulfilled. Note that at the end of letting-everyone-know stage, all agents have synchronized knowledge of statuses of π.
- According to their definitions, *ProcCommunityApproval* and *ProcCommunityRefusal* are mutual exclusive for an agent to act. Since at this stage agents know the same statuses of π then agents determine the same conclusion for π. This fulfills $E_C \forall_{\geq m} i \in C : \lambda_i \pi \implies \lambda_C \pi$ (axiom 3.6 which superimposes axiom 3.5).
- According to definition of *ProcCommunityApproval* when majority of agents approve the proposal then each agent updates *ifacts* such that $ifacts' = ifacts \backslash P_o \cup P_n$. According to definition of *ProcCommunityRefusal*, when approvals do not sufficient to represents majority of agents then *ifacts* do not change such that $ifacts' = ifacts$. That is, the definition of *ProcCommunityApproval* fulfills the definition of $\lambda_C \pi$ (equation 3.7). □

Transaction Cost and Scalability

Additionally, the transaction algorithm presented here requires agents to perform broadcast approval / refusal statements in order to complete transaction. This makes communication cost relative high. One transaction requires $n^2 - 2n$ message transmissions. Indeed, this scheme is not considerably efficient to be used in practice. In addition, the transaction algorithm assumes that agents are capable to maintain and synchronize list of accepted institutional-facts. This would be scalability issue as the community grows. Therefore, chapter 6 will present transaction protocols developed from the algorithm utilizing distributed cryptography. The system reduces communication cost as well as eliminates the scalability problem.

3.5 Summary

This chapter delivered answers of two questions stated in the beginning of the chapter, namely to build system of accounts which is based on the institutional facts, and to design accounting mechanism that based on the developed system of accounts.

Fundamentals axioms of creation and elimination of institutional facts were presented to establish the system of accounts. Fundamental axioms of approval of transaction proposal was also presented intended to be the bridge of system accounts to the establishment of the accounting mechanism which is based on it.

Transaction based on alteration of states of finances and transaction based on exchange of money, were presented to exemplify schemes of transactions that can be applied using developed transaction system.

The last part of the chapter presented generic transaction algorithm specified in TLA+ to capture as well as to express actions of agents in initiating transaction and the collaboration actions to complete it. An predicate action diagram of corresponding specification was presented to aid explanation of the algorithm. At last, the algorithm is proven to fulfill fundamental axioms presented in the beginning of the chapter.

Chapter 4

Collective Authorization and Social Control

Mechanism to complete transaction presented in previous chapter introduces a mechanism that enables community to govern transactions occured in individual level. This chapter takes advantage this in developing collective authorization scheme that enables agents to collectively approve or forbid transactions reasoned by agent's reputation.

This chapter begins by presenting the backgrounds providing the reason and the notions behind the concept. Next, vote based collective authorization scheme is presented in which reputation of the seller as well as transaction profile signify community decisions. The subsequent section presents collective authorization algorithm that completes the specification of algorithm presented in previous chapter.

4.1 Backgrounds

4.1.1 Importance of Trust

The essential concept of the framework of communally governed transactions is collaboration as part of self-organization. Without collaboration, community would not be able to deliver its function in completing transactions. Therefore, it is important to maintain factors that sustains the collaboration. Some of those factors are altruism, voluntarism, and trust. The first two are the matter of how useful and important the community is to its members. They belong to the puzzle which is not discussed here.

Trust has been known to be the magnitude ingredients in establishing and adhering collaboration [Luhmann, 1979] [Buskens, 1999] [Stompzka, 1999]. In the context of online trading, to where proposed framework is aimed, the perturbation of trust can be caused from disappointment in receiving unwanted outcome of transaction.

Unwanted outcome means that buyer perceives that seller had delivered bad quality of goods or services. In other words, the buyer had made a transaction with the seller who performed improperly in the transaction.

If this happens frequently then distrust is began to spread. It might cause the turbulence of collaboration. Hence, a mechanism should be designed in the framework that emerges a way in preventing possible unwanted outcomes i.e. preventing buyer to meet improper seller. Since no authority exists in the system to perform this function, the design should form a self-control mechanism to filter bad behavior and thus encouraging good behavior among agents

4.1.2 Social Control and Reputation

Social control is social mechanism that regulates individual behavior in a social system to conform with local rules or norms. The implementation of social control is often seen in traditional society where norms are significantly influential and hence strongly enforced. Contrary to the formal law, social control is exercised collectively by local people, not by the government.

The example of social control is neighborhood watch, a collective effort in conducting security control around the village or neighborhood. It is mainly self-managed by local people and received minor influence from the government. The objective is not only that the village would become secure but also that any kind of behavior which might violate local norms such as adultery or gambling can be prevented. This kind of collective action reveals **social filtering mechanism** in which people collectively filter possible behaviors which might violate the norms.

Another form of social control is **social punishment** or **social sanction**. Those who have violated the norms are collectively punished or sanctioned. The implementation of punishment varies from isolation, discrimination, social shaming e.g. processioned through the village, to the exclusion from society. Provided with those examples, others learn that it is bad to violate the norms. Thus, it encourages good behavior among the people.

Human beings have developed the concept of trust and reputation. Should one face dilemma whether to place trust, one collects for information about the subject of trust by which one can learn and conclude his reputation. Given subject's reputation as well as the risk, one has more complete information to be able to decide whether to place trust or not.

The use of reputation in every decision trust emerges **social filtering mechanism.** This mechanism limits one's behavior based on his reputation. Those who have high reputation are likely to receive trust more often than those who don't. In this respect, it emerges **social sanctioning mechanism**. That is, persons who have bad reputation are seldom being trusted for any kind of interactions and thus collectively isolated from the society.

4.1.3 Design Goal

Based on the description above, the goal of the design presented in this chapter is to develop mechanism that enable social control. The mechanism should be embedded into the proposed framework that enables the agents to collectively filter possible unwanted outcome of transactions that involve dishonest sellers. Here, agents takes seller reputation as input to the filter in order to collectively decide whether to approve or to forbid the transaction, i.e. to trust the seller or not. This mechanism emerges collective authorization service which is also protective measure for the buyers against sellers with bad reputation.

4.2 Reputation System

To provide common understanding of the reputation system, it is important to comprehendly discuss reputation system as decision support system in online communities.

4.2.1 Notions of Trust and Reputation

Recalling the the game of trust discussed in chapter 2, the person who faces trust dilemma is denoted with *trustor*, and the person who is going to be trusted is denoted with *trustee*. In trading community, trustor is identic with the buyer who pays an amount of money to the seller, the trustee, who is expected to deliver goods or services.

The nature of trading in online environment introduces time lag between the payment and the outcome. One is unable to know the quality of goods or services in advance. Hence, trustor has to make decision to rely his investment on the reliability of trustee. This decision requires certain level of confidence that that trustee will deliver resources as expected in return to the payment. This level of confidence can be reached when trustor has sufficient and complete information about trustee's behavior from which he can conclude reliability of trustee.

The following definitions are crystallized from [Gambetta, 1988] [Jøsang et al., 2005] who describe the notions of trust and reputation in online communities.

Definition 4.1: *Reliability trust is subjective probability by which trustor expects trustee to perform intended action on which trustor's welfare depends.*

Reliability trust is usually derived from personal experiences which in many situations are considered as incomplete, unless trustor knows trustee very well. In order to have near complete information, trustor gathers information from others. This information exhibits others' opinions about the trustee: their recommendations.

Definition 4.2: *Recommendation is what other say about someone or something.*

Recommender is the third person who gives his opinion about the trustee. Hence, recommendation is a second hand trust referrals. Information gathered from all possible recommender concludes trustee's reputation.

Definition 4.3: *Reputation is what is generally being said about someone or something.*

Reputation represents the whole opinion from the community towards trustworthiness of trustee. That is, it represents a state and the wholeness. Its difference with recommendations is that recommendation represents the process of informing rather than state.

Provided with trustee's reputation, trustor evaluates reliability trust of trustee in order to make a final decision to place trust.

Definition 4.4: *Decision trust is the extent to which trustor is willing to put his welfare on trustee's authority despite of negative consequences trustor might receive.*

That is, after deriving reputation from trustee, trustor considers the risk he might suffer when he places the trust and trustee abuses the trust, as well as the advantage that trustor might gain when he places the trust and trustee honors it.

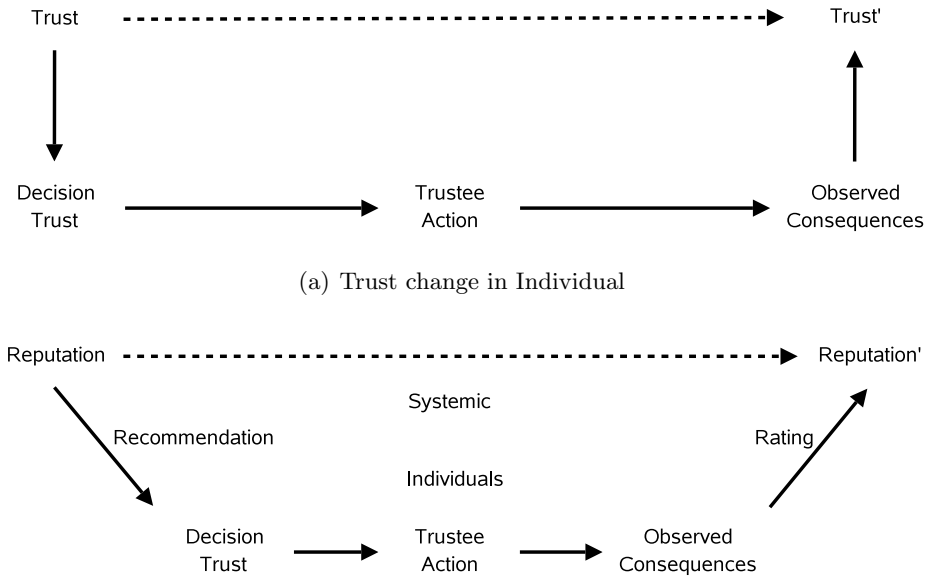
4.2.2 How Reputation Changes

How trust and reputation evolves ? Figure 4.1 presents typical change of trust in individual level and the change of reputation in societal level. Here, trust and reputation represent the state of trustworthiness of a trustee as seen from individuals and from societal level as a whole, respectively.

Trust or reputation of trustee is used as parameter in decision trust. Positive decision opens an opportunity for trustee to act and conversely negative decision blocks it. Given opportunity to act, trustee performs according to his behavior from which trustor receives consequences. Observing the consequences, trustor updates his level of trust towards the trustee. Negative consequence should lower the level of trust and conversely positive consequence should raise the level of trust. When trustor makes his opinion known to the community, then the propagation changes trustee's reputation in the community.

4.2.3 Elements of Reputation System

A reputation system as decision trust support system has three elements: how reputation is propagated, how reputation is expressed and computed, and how decision trust is made based on the reputation. Figure 4.2 illustrates elements of reputation system.



(a) Trust change in Individual

(b) Reputation Change

Figure 4.1: The Change of Trust and Reputation

Reputation propagation

Recommendation is the natural way known in everyday life to conclude reliability trust of the trustee. Josang et. al. [Jøsang et al., 2005] entitled the collaboration system which utilizes recommendation to filter potential bad behavior with *Collaboration Filtering System* (CF). It is presumed in this system that everyone have different trusting behavior. Different taste of trust means that it is natural when one puts his personal experience in higher weight in the consideration.

In CF, propagation of reputation is commonly **initiated by the trustor** by querying anyone who have previous experience with the trustee to gather their recommendations.

CF's counter-part, a so-called *Collaboration Sanctioning System* (CS), on the other hand, presumes that everyone has the same 'taste' of trust and everyone tells his opinion to anyone else about experience with the trustee. This opinion is called *rating* which is communicated by trustor after consequences is received and observed.

According to Josang et. al. the nature of CS is suitable to be used in centralized system where trusted third party arranges the collection and the presentation of the rating, whereas CF is suitable for decentralized system in which such entity does not exist. One can make hybrid system utilizing both advantages, though.

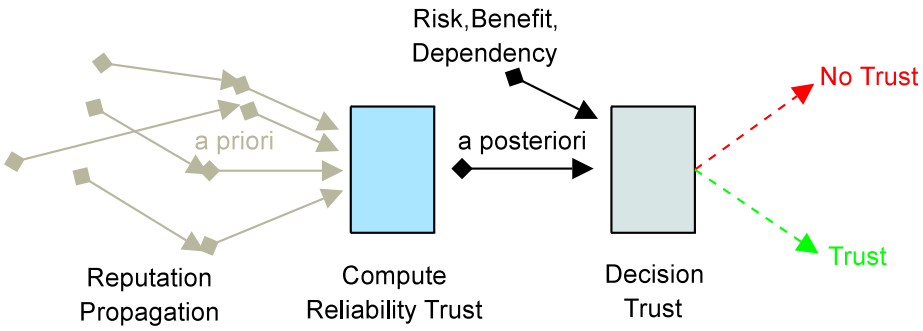


Figure 4.2: Elements of Reputation System

Method to compute reputation

Rating or recommendation can be presented and computed in several ways. The main goal is to compute *a posteriori* reputation from *a priori* ratings, opinions, or recommendations. As explained in chapter 2, there are known trust computation methods that have been studied, e.g. simple average, Bayesian system, and discrete system.

Simple average method takes numerical scores from trustee's ratings from which the sum are divided by the number of experiences. Bayesian counterpart takes binary input in that 1 represents good experience and 0 represents bad experience from which trustor computes reputation of trustee using Bayesian algorithm. In discrete system, opinion or rating of trustee is expressed with discrete values e.g. (Very Good, Good, Average, Bad, Very Bad). Although the expression is very natural, in certain circumstances, it is difficult to make decision trust when risk should be taken into account.

Method to reason decision trust

Given reputation state of trustee, one should have method to assess the parameter and to decide whether to place trust or not. In discrete system and simple average, for example, one can set a threshold value from which one decides to place trust. When trustee's reputation is above the threshold, one places trust, whereas the value is below the threshold, trustee does not earn trust.

Other methods take the value of forthcoming risk and benefit as parameters in decision trust. Josang and Presti [Jøsang and Presti, 2004] as well as Coleman [Coleman, 1990] formulate methods to deal with decision trust that take those parameters into account. Both are based on rational comparison between ratio of potential benefit for the trustor when trustee honors the trust, and potential loss or risk when trustee misplaces the trust.

4.3 The Design of Collective Authorization

There are two options that can be implemented in sustaining trust within community. The first is to employ decentralized recommendation system in which decision trust is made in ad-hoc manner by the trustor himself. Provided with propagated trustee's reputation, trustor enables to make decision trust independently based on his own will and considerations. The system forms decision support system for the agents. This scheme is not new and has been implemented in most of virtual communities. The effectiveness of this system is questionable, since the trustor is allowed to make any decision that might allow untrustworthy trustee to have transaction.

The second option is more interesting to study: a mechanism that enables community to make collective decision trust based on trustee reputation. It allows community to govern and secure transactions in the individual level. In other words, community can collectively authorize or not authorize a transaction based on seller's reputation.

4.3.1 Basic System Setup

Recalling transaction algorithm presented in the previous chapter. First an encounter occurs between buyer and seller who propose transaction proposal to the community. Each agent express his positive or negative statement of proposal approval based on certain reasons e.g. validity of transaction proposal. If positive statements are satisfactory in representing the majority of community, then transaction proposal is approved and thus transaction proceeds.

Collective authorization is embedded into the framework by introducing seller reputation in the reasoning. Positive statement from an agent represents partial decision trust from the whole collective decision trust. When majority of agents express positive statement regarding the transaction then majority of community trusts the seller to conduct the transaction and thus the transaction is collectively authorized. Conversely, when the seller receives less than adequate positive statements, then the seller is not trusted to do the transaction, e.g. transaction is aborted or collectively not authorized. In this scheme, collaborative agents in the community act as a intermediate trustor for the buyer. Hence, in this setting:

1. Buyer is the requester for community decision trust
2. Seller acts as trustee who also requests for community decision trust
3. Other agents act as intermediate trustor on behalf of the buyer

If a transaction is collectively authorized or approved, then transaction occurs. After delivery of trading objects, buyer assesses the outcome of transaction and makes his

opinion or rating of the outcome of transaction known to community. Based on this information community alters seller's reputation in their knowledge.

Essentially, the design of collective authorization is distribution of decision trust to the community. It forms collaboration filtering scheme to prevent possible unwanted outcome. It also forms collaboration sanctioning scheme, in which community is able to block transaction proposal involving seller with bad reputation.

Following sections shall go further into detail how partial decision trust are made as well as how ratings are propagated.

4.3.2 Community Decision Trust

Every transaction entails forthcoming risk as well as benefit, the transaction profile that buyer perceives. Hence, the challenge in the design of collective decision trust is how each partial decision trust captures transaction profile. To decide trust based on predefined threshold value shall not capture what buyer perceives as transaction profile.

Coleman's model of decision trust is based on rational decision making. It captures potential risk and potential gain from a transaction. Let $\rho \in [0, 1]$ be opinion of the trustor toward probability that trustee will honor the trust (reliability trust of trustee), $L \in \mathbb{R}$ be potential loss or risk of the transaction when trustee misplaces the trust, $G \in \mathbb{R}$ be potential gain of the transaction when trustee honors the trust, and $LG = \frac{L}{G}$ is the loss to gain ratio. Decision trust is based on rational thinking such that trustor will place trust when potential gain of the transaction given probability that trustee will honors $G \times \rho$ is greater than potential loss of the transaction given probability that trustee will misplace the trust $L \times (1 - \rho)$. Formally:

$$\text{Decision Trust} = \begin{cases} \text{Place Trust} & \text{if } \frac{\rho}{1-\rho} > LG \\ \text{No Trust} & \text{if } \frac{\rho}{1-\rho} < LG \\ \text{Indifferent} & \text{if } \frac{\rho}{1-\rho} = LG \end{cases} \tag{4.1}$$

Coleman's decision trust model is able to capture the ratio of potential risk and gain of the transaction as well as trustee reputation. Therefore, in every transaction proposal, buyer shall inform the transaction profile LG value to be used by intermediate trustors as parameter for their partial decision trust.

Let $\rho \in \Omega = C \times C \times [0, 1]$ be opinions of all agents in community C towards reliability trust of another. Let $\Omega_s \subset \Omega$ be set of opinions of all agents about probability that agent $s \in C$ will honor the trust, $\rho_{as} \in \Omega_s$ be agent a opinion about agent s, and $LG \in \mathbb{R}$ is loss to gain ratio that the buyer expects from the output of a trading or a transaction. Function $PTrust : \Omega \times \mathbb{R} \rightarrow \{1, 0\}$ is introduced to express partial decision trust by an agent, where 1 represents "Place Trust" and 0 represents "No Trust". Partial decision trust by agent a of transaction involving

agent s as seller in which the buyer perceives loss to gain ratio value LG, is based on the following condition:

$$PTrust(\rho_{as}, LG) = \begin{cases} 1 & \text{if } \frac{\rho_{as}}{1-\rho_{as}} > LG \\ 0 & \text{if } \frac{\rho_{as}}{1-\rho_{as}} \leq LG \end{cases} \tag{4.2}$$

Community decision trust is the product of all partial decisions. It can be the result of long and complex consensus or it can be as simple as the result of simple voting that is employed here utilizing transaction approval and refusal scheme presented in previous chapter.

Thus, community decision trust $Trust : \Omega^n \times \mathbb{R} \rightarrow \{1, 0\}$ on a transaction in which s is the seller and the buyer perceives transaction profile as LG, is based on the following condition:

$$Trust(\Omega_s, LG) = \begin{cases} 1 & \text{if } \sum_{a \in C} PTrust(\rho_{as}, LG) \geq m \\ 0 & \text{if } \sum_{a \in C} PTrust(\rho_{as}, LG) < m \end{cases} \tag{4.3}$$

As specified before, m is number of agents which is satisfactory to represent majority of community. The domain of function *Trust* is binary value $\{0, 1\}$ which in this respect 1 represents "*Transaction is collectively Authorized*", and 0 represents "*Transaction is not collectively authorized*".

4.3.3 Reputation Propagation and Concluding Reliability Trust

Coleman's model of decision trust considers the probability of behavior of trustee in a transaction. The value of probability, in this respect, is subjective opinion of the trustor towards reliability of the trustee. It is derived from personal experiences as well as what other say about the trustee: his reputation. Here, there are two important components in the scheme, namely (i) way to propagate recommendations or ratings, and (ii) method how to conclude or to compute the probability value from reputation.

To achieve reasonable performance, collective decision trust should happen immediately. Inevitably, there is no space to interact in advance for exchanging or querying recommendations or opinions. Hence, ratings of an outcome of transaction should be made known to public after transaction completes, in particular after the buyer receives goods or services from the seller. The received ratings alters agent's opinion about trustworthiness of seller.

How agent's opinion is altered depends on the method used to compute reputation. Previously, there were discussion about methods to compute reputation, for example, Simple average, Bayesian system, and Discrete system. These method can be employed into the framework as long as it can be perceived as "probability"

$\{0, 1\}$. The simplest method is to directly use percentage scheme. For example, let r be the value of trustee's reputation and $rmax$ be the maximum value of reputation in particular system. The probability that trustee will honor the trust is $\frac{r}{rmax} \times 100$ percent.

Using this point of view, it is relative easy to conclude the reliability trust of trustee using method such as simple average or weighted values. Even in discrete system, the values of $\langle$Very Good, Good ,Average, Bad, Very Bad$\rangle$ can be converted to $\langle 1, 0.75, 0.5, 0.25, 0\rangle$ counterparts. So when one consider that trustee has Good reputation then his opinion on the probability that trustee will honor the trust is 75% or 0.75.

Note that this method is simplification of translation from one system to another. Additionally, the study of semantic translation between computational should be conducted. Nevertheless, it is beyond the discussion in this thesis.

The Beta Reputation

Algorithm presented in this chapter employs Beta reputation system proposed by [Ismail and Jøsang, 2002]. The reason to use Beta reputation is merely example. It is simple and the output of computation is presented as probability. Thus, it can be easily embedded into collective decision trust scheme. Nevertheless, any other methods can also be employed in the system under specific considerations using the translation method previously discussed in 4.3.3.

In Beta reputation system, rating from outcome of transactions as perceived by the rater is expressed as binary values; 0 for bad result and 1 for good result. Rating is propagated by the buyer after he receives outcome from the seller to the rest of community. Each agent who receives this rating updates his knowledge about reputation of the particular seller represented in tuple $\langle sum\_good, sum\_bad\rangle$ where sum_good is the sum of good results and sum_bad is sum of bad results.

Beta reputation system concludes the expectation or probability that a trustee or seller will honor the trust with:

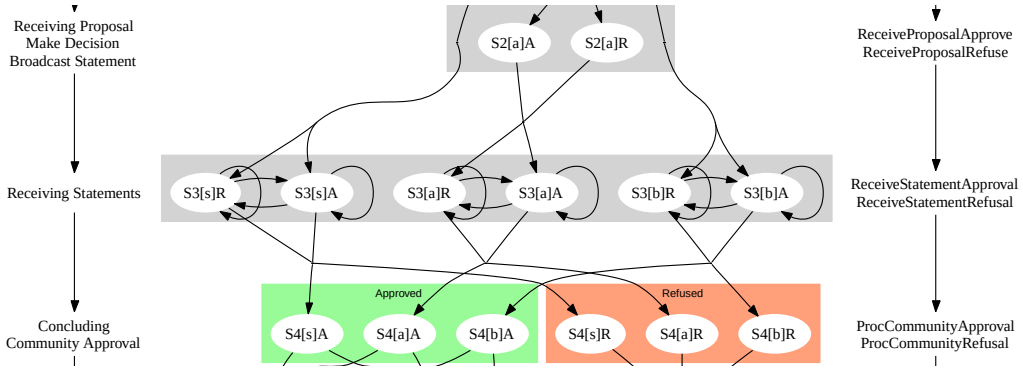
$$\rho = \frac{sum\_good + 1}{sum\_good + sum\_bad + 2} \tag{4.4}$$

Using Beta reputation, an agent simply maintains the sum_good and sum_bad values of all agents in community accumulated from his own experiences as well as ratings received from others. The next section will explain collective authorization algorithm employing Beta reputation in detail.

4.4 Collective Authorization Algorithm in TLA+

4.4.1 Re-Introduction

Some of the materials in this discussion refer to the transaction algorithm as presented in previous chapter. The following part of state machine recalls collective approval and refusal of a transaction proposal:



In receiving a transaction proposal, an agent decides whether he approves or refuses a transaction proposal determined by *ReceiveProposalApprove* and *ReceiveProposalRefuse*. The statements of approval and refusal are made public from which each agent counts statements of others. At the end, all agents are in the same state in that they know whether agents approves or refuses transaction based on predefined threshold representing majority. In the algorithm, this is determined by *ProcCommunityApproval* and *ProcCommunityRefusal*.

In this respect, the reason to approve or refuse the proposal is merely based on validity of the proposal. This chapter advances this scheme further by incorporating partial trust decision (eq 4.2) in the reasoning of *ReceiveProposalApprove* and *ReceiveProposalRefuse*. In this perspective, the semantic of the words *Approve*, *Refuse*, *Approval*, *Refusal* are perceived as *Place Trust*, *No Trust*, *Authorized*, and *Unauthorized*, respectively.

Reputation Storage

Beta reputation considers the sum of good experiences as well as the sum of bad experiences. Recall table 3.1 showing records of internal knowledge of agent. An agent stores reputation of all agents in the following record:

$$experience = community \rightarrow (sum\_good, sum\_bad)$$

Both *sum_good* and *sum_bad* are integer greater or equal than zero. In TLA, they are defined as *Naturals*. These two values will be altered when agent receives

rating of outcome of transaction and will be used in the partial decision trust when a transaction proposal is received.

4.4.2 Collective Authorization

In a transaction proposal, buyer and seller put information about transaction loss to gain ratio named with $lgratio \in \mathbb{R}$. In receiving transaction proposal, agent takes *lgratio* as well as seller reputation in the reasoning of his partial decision trust which in the algorithm is determined with *ReceiveProposalApprove* and *ReceiveProposalRefuse*.

As remainder the following descriptions are taken from previous chapter:

$S_0 \rightarrow S_2[a]A$ **ReceiveProposalApprove**
Action *ReceiveProposalApprove* is enabled if an agent receives a message containing a proposal, and both *isPropValid* and *trustDecision* are *TRUE*

$S_0 \rightarrow S_2[a]R$ **ReceiveProposalRefuse**
Action *ReceiveProposalApprove* is enabled an agent receives a message containing a proposal, and either *isPropValid* and *trustDecision* is *FALSE*

Discussion in this chapter assumes that *isPropValid* is always true which implies approval and refusal is based merely on the result of function *trustDecision*. That is, if *trustDecision* is true then *ReceiveProposalApprove* is enabled whereas if *trustDecision* is false then *ReceiveProposalRefuse* is enabled, but not both on the same transaction proposal.

Function $trustDecision : \mathbb{R} \times \mathbb{Z}^2 \rightarrow \mathbb{B}$ reasons partial decision trust of a transaction proposal involving particular seller. In TLA, *trustDecision* is defined as:

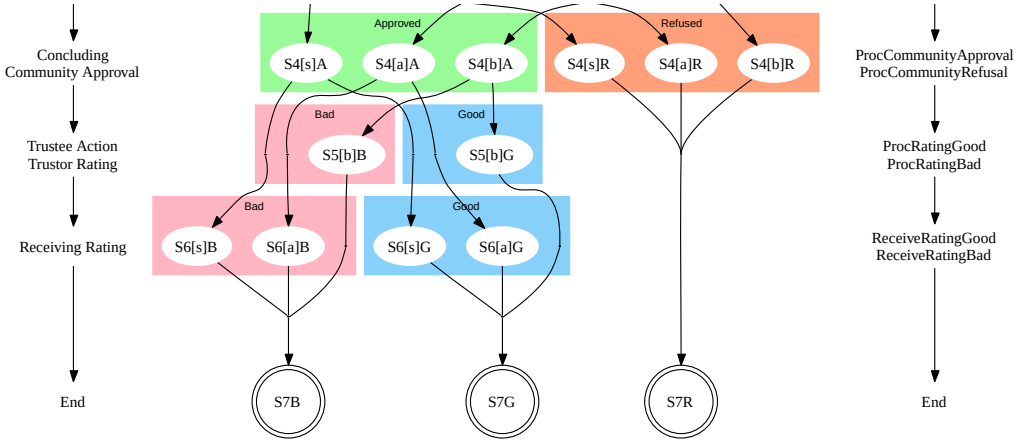
$$
\begin{array}{l}
trustDecision \triangleq \\
\quad [lg \in Real, \\
\quad\ sum\_good \in Nat, \\
\quad\ sum\_bad \in Nat \\
\qquad \mapsto \text{LET } \rho \triangleq \\
\qquad\qquad (sum\_good + 1)/(sum\_good + sum\_bad + 2) \\
\qquad\quad \text{IN} \quad (\text{IF } \rho > (1 - \rho) * lg \\
\qquad\qquad\quad \text{THEN TRUE} \\
\qquad\qquad\quad \text{ELSE FALSE}) \\
\quad]
\end{array}
$$

This function uses Beta reputation (equation 4.4) to compute reliability trust of seller from which the result is used to determine partial decision trust as specified in equation 4.2.

In the protocol run, an agent approves the transaction proposal if he, as part of intermediate trustors, decides to place trust (*trustDecision*=TRUE) on the seller to conduct proposed transaction. Conversely, he refuses the transaction proposal. Each intermediate trustor broadcast the statement containing his part of decision trust to the rest of community. After receiving all statements, an agent is able to determine whether community decides to collectively place trust on the seller and thus authorizes the proposed transaction, or not to place trust on the seller and thus does not authorize proposed transaction. As previously discussed, this is determined by *ProcCommunityApproval* and *ProcCommunityRefusal* based on the majority voting. Thus, this step is in accordance with equation 4.3.

4.4.3 Rating Propagation

The following discussion refers to the table 4.1 as well as following part of predicate action diagram:



Consider that transaction is authorized or approved ($S_4[s]A$, $S_4[a]A$, $S_4[b]A$). Buyer and seller are permitted to conduct the trading or the exchange. Seller delivers trading objects or services to the buyer (or does opposite action if seller is untrustworthy) followed by buyer assessment on the outcome of transaction. Buyer sets the rating of the outcome of transaction consists of two possibilities. If the buyer satisfies with the outcome, then he gives good rating (1). On the contrary, if the outcome of the transaction is not satisfactory, then he gives bad rating (0). This rating is broadcasted to the rest of community. In the algorithm this action is determined by *ProcRatingGood* and *ProcRatingBad*.

$S_4[b]A \rightarrow S_5[b]G$ **ProcRatingGood**

Action *ProcRatingGood*, implying this particular state transition, is enabled if an agent finds a proposal π that is already approved ($\pi.decided = TRUE$),

State Transition	Pre Condition	**Action** / Description	Post Condition
$S_4[b]A \rightarrow S_5[b]G$ $b \in$ *Community*	agent[b].proposals[π]. proposed=FALSE decided=TRUE rated=FALSE buyer=b *rateTransResult*[π]=Good agent[b].experience[π.seller] sum_good sum_bad	**ProcRatingGood** *Agent finds completed transaction from proposal π in which he was the buyer, he has not rate the transaction, and the outcome of transaction is considered to be good. Agent updates his knowledge about reputation of the seller π.seller, marks π as rated, and broadcast his statement to other.*	agent[b].proposals[π]. <proposed> <decided> rated=TRUE agent[b].experience[π.seller] sum_good+=1 <sum_bad>
$S_4[b]A \rightarrow S_5[b]B$ $b \in$ *Community*	agent[b].proposals[π]. proposed=FALSE decided=TRUE rated=FALSE buyer=b *rateTransResult*[π]=Bad agent[b].experience[π.seller] sum_good sum_bad	**ProcRatingBad** *Agent finds completed transaction from proposal π in which he was the buyer, he has not rate the transaction, and the outcome of transaction is considered to be bad. Agent updates his knowledge about reputation of the seller π.seller, marks π as rated, and broadcast his statement to other.*	agent[b].proposals[π]. <proposed> <decided> rated=TRUE agent[b].experience[π.seller] <sum_good> sum_bad+=1
$S_4[s]A \rightarrow S_6[s]G$ $S_4[a]A \rightarrow S_6[a]G$ $a, s \in$ *Community* $a \neq s$	agent[{a,s}].proposals[π]. proposed=FALSE decided=TRUE rated=FALSE agent[a|s].experience[π.seller] sum_good sum_bad	**ReceiveRatingGood** *Agent receives Good rating from buyer b regarding the outcome of transaction that was proposed with proposal π. Agent marks π as rated and alters seller reputation in his knowledge by adding one to the sum_good*	agent[{a,s}].proposals[π]. <proposed> <decided> rated=TRUE agent[b].experience[π.seller] sum_good+=1 <sum_bad>
$S_4[s]A \rightarrow S_6[s]B$ $S_4[a]A \rightarrow S_6[a]B$ $a, s \in$ *Community* $a \neq s$	agent[{a,s}].proposals[π]. proposed=FALSE decided=TRUE rated=FALSE agent[a|s].experience[π.seller] sum_good sum_bad	**ReceiveRatingBad** *Agent receives Bad rating from buyer b regarding the outcome of transaction that was proposed with proposal π. Agent marks π as rated and alters seller reputation in his knowledge by adding one to the sum_good*	agent[{a,s}].proposals[π]. <proposed> <decided> rated=TRUE agent[b].experience[π.seller] <sum_good> sum_bad+=1

Description
The < > bracket indicates that the value of variable in the bracket does not change in post condition
The += operator has the same semantic as "increment by" operator += used in C language
Irrelevant variables are not presented.

Table 4.1: State Transitions (Part III)

not yet rated ($\pi.rated = FALSE$), he was buyer in the proposal ($\pi.buyer = a$), and he finds that in this transaction the seller delivered good outcome ($rateTransResult = good$). Agent alters his knowledge that he had good experience with particular seller by adding 1 to the $expirience[\pi.seller].sum\_good$ and marks the proposal as rated. Finally, he broadcast the rating to the rest

of community.

$S_4[b]A \rightarrow S_5[b]B$ **ProcRatingBad**
This action is enabled if an agent finds a proposal π that is already approved ($\pi.decided = TRUE$), not yet rated ($\pi.rated = FALSE$), he was buyer in the proposal ($\pi.buyer = a$), and he finds that in this transaction the seller delivered bad outcome ($rateTransResult = bad$). Agent alters his knowledge that he had bad experience with particular seller by adding 1 to the $expirience[\pi.seller].sum\_bad$ and marks the proposal as rated. Finally, he broadcast the rating to the rest of community.

Note that the algorithm does not determine whether outcome of transaction is good or bad. Function $rateTransResult : p\_proposal \rightarrow \{good, bad\}$ is there to arbitrary express the result of buyer judgment of the outcome of transaction which should be good or bad, but not both.

Further, each agent $a \in C, a \neq b$ receives rating from the buyer b and alters his knowledge accordingly. This is determined by *ReceiveRatingGood* and *ReceiveRatingBad*.

$S_4[a|s]A \rightarrow S_6[a|s]G$ **ReceiveRatingGood**
This action is enabled if an agent finds a message in the first sequence *msgfifo* containing rating information stating good outcome of a transaction (originally from proposal π and $\pi.decided = TRUE$) that is not yet rated ($\pi.rated = FALSE$). Agent alters his knowledge that the buyer had good experience with particular seller by adding 1 to the $expirience[\pi.seller].sum\_good$ and marks the proposal as rated.

$S_4[a|s]A \rightarrow S_6[a|s]B$ **ReceiveRatingBad**
This action is enabled if an agent $a \in Community$ finds a message in the first sequence *msgfifo* containing rating information stating bad outcome of a transaction (originally from proposal π and $\pi.decided = TRUE$) that is not yet rated ($\pi.rated = FALSE$). Agent alters his knowledge that the buyer had bad experience with particular seller by adding 1 to the $expirience[\pi.seller].sum\_bad$ and marks the proposal as rated.

In this stage of transaction, all agents should have altered their knowledge about s's reputation according to the rating that buyer has given.

4.4.4 Transaction Summary

Figure 4.3 illustrates the summary of transaction steps according the the transaction algorithm. First, encounter (S_0) between buyer and seller occurs followed by

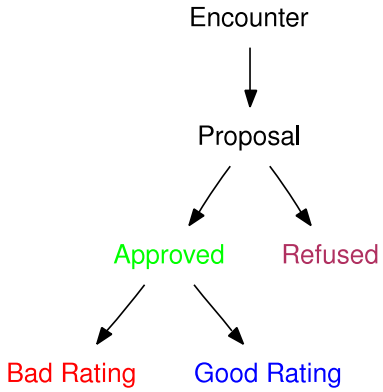


Figure 4.3: Summary of Completion of Transaction

proposing transaction proposal (S_1). Agents collectively decide whether to approve the proposal or to refuse it ($S_2 \rightarrow S_3 \rightarrow S_4$). As discussed in previous chapter, when the proposal is refused, system variables are "roll-backed" so that S_7R is the same as S_0. On the other hand, if the proposal is approved then transaction "commits" and transacting agents are allowed to trade. If the outcome is good then buyer broadcast good rating (S_5G) from which others alters their knowledge (S_6G) that brings overall system state to S_7G. In contrast, if bad rating is broadcasted (S_5B) and received by other (S_6B) then it bring the system to the state S_7B.

4.5 Summary

This chapter presents a mechanism of collective authorization with desire to let agents self-organize collective filtering to prevent risky transactions to occur. The collective decision is based on reputation of seller as well as risk to gain ratio of individual transaction. Reputation of the seller is altered when the buyer lets other know his opinion of completed transaction. The final section of this chapter presents an algorithm of collective authorization which is embedded into transaction algorithm that was previously presented in chapter 3.

Chapter 5

Simulation on Collective Authorization

Chapter 4 presented collective authorization that is designed with a wish that social control can be established. This chapter studies the scheme further using simulation. First, it introduces the underlying idea of the simulation system as well as develops simulation algorithm. The subsequent section describes the utilization of simulator in conducting set of experiments to study how the system evolve in performing collective authorization under various initial conditions. Finally, the results of experiment are presented and discussed.

5.1 Underlying Concept of Simulation on Social Control

5.1.1 Background and Objective

The work in this thesis can be roughly devided into two parts. The first part is the design of transaction algorithm to accomplish transaction among decentralized trading agents. The second part is to develop collective authorization reasoned by reputation to establish social control and to maintain trust as well. This chapter is about the study of the last mentioned part. Since proposed framework has not been implemented in real world, it is important to know in advance how social control is established and under which conditions it works.

Simulation is a method to understand the model [Gilbert and Troitzsch, 1999]. By applying collective authorization algorithm on sets of simulated data, behavior of the system can be observed and studied. Here, simulation environment are utilized to conduct series of experiments under arbitrary parameters.

The objective of this simulation is twofold (i) as a preliminary study of how social control works on the system of agents applying collective authorization algorithm, and (ii) to take advantage of simulator to conduct experiments to observe how system

Behavior Type	Possible Outcome
Honest	80-100
Fair	40-80
Unpredictable	0-100
Dishonest	0-40

Table 5.1: Range of Possible Transaction Outcomes of each Behavior Type

might evolve from arbitrary initial conditions to the desire state - trust.

Collective authorization algorithm uses voting scheme to achieve collective decision trust as well as broadcast rating in order to propagate trust information. This scheme differs from known methods that are used in distributed recommendation system or by centralized rating provider. Based on this background, recommendation system is also incorporated into the simulator for the purpose of comparison.

5.1.2 Natural Selection

In order to observe evolution of the system, simulation mimics natural selection process. Every agent possesses an amount of energy, the fitness, from which the existence of the agent in social engagement is fueled. At each encounter with a buyer, the fitness of seller decreases by a constant predefined value. If transaction is authorized, then seller receives an incentive that increases his fitness by a number of predefined value. Conversely, seller gets nothing to increase his fitness. Latter, if an agent runs out of fitness, he can not engage in social interaction. That is, he dies.

In the simulation, there are four types of agents arbitrarily set to exhibit their behaviors as seller in transaction, namely, **Honest**, **Unpredictable**, **Fair**, and **Dishonest**. Each of the behavior type represents set of possible outcomes of transaction from which seller of the corresponding type produces in having a transaction. According to their assigned names, Honest produces good outcome. On the other hand, Unpredictable, Fair, and Dishonest produce unpredictable outcome, fair outcome, and bad outcome, respectively. Table 5.1 lists possible outcomes from each behavior types expressed in numerical value from 0 to 100. Moreover, this simulation assumes that buyer is always honest in giving the rating. Thus, the outcome of transaction is automatically set as the rating of the transaction itself.

The scheme of collective authorization promotes states that good behaving agents are likely authorized to do transaction, and conversely bad behaving agents are unlikely authorized to do transaction. By incorporating natural selection in the simulation, one could observe how population of each behavior types of agents evolves. The dynamics of populations are used as indicator for measuring how the algorithm works under various conditions. As mentioned in previous chapter, turbulence of co-

operation can be the result of frequently unwanted outcomes performed by dishonest agents. Thus, one looks for conditions of the system where dishonests are socially filtered implying population of Dishonests as well as Unpredictables are reduced as soon as possible, but Honests are still preserved. This property is defined in this simulation study as sustainability of trust.

5.1.3 Simulation Parameters

In collective authorization algorithm, partial decision trust is essentially made using two parameters: the loss to gain ratio (LG) which is a measure of risk and gain of transaction and expectation that the seller will behave accordingly in the transaction (ρ).

Distribution of ρ among agents represents the level of trust in community. It pictures how agents trust one another. Low values in the distribution of ρ represent low level of trust. High values in the distribution of ρ represent high level of trust among agents. Distribution of ρ change as transactions were conducted and ratings were aggregated.

Loss to gain ratio LG represents transaction profile in respect to risk and benefit. Transaction profile explains type of goods or services being exchanged in the community. Generally, trading community exists from agents which have the same interest, e.g. MP3 trading community or Computer Hardware trading community. In this perspective, each community has a transaction profile. Buying MP3, for example, is kind of transaction that relatively has the same level of loss and benefit, hence LG is around 1. On the other hand, trading expensive goods such as high-end laptops or providing financial service such as loan is considered to be risky, hence, the value of LG is high. The example for low LG transaction would be buying lottery. For lottery fans, buying lottery ticket would have LG value less than 1. For them lottery ticket is cheap and has high potential profit.

The simulation in this chapter concerns with initial distribution of ρ as well as transaction profile LG to be used as main parameter in the simulation. Initial distribution of ρ represents initial level of trust when community if formed. LG represents what type of good or services being exchanged in the community. Here, every combination of parameter is simulated to see which combination maintains the property of trust in the simulated system.

5.2 Simulator Program

Simulator is programmed using *NetLogo* version 3.1.3, a popular programmable modeling environment developed by [Wilensky, 1999, Wilensky and Stroup, 1999]. NetLogo is an environment for simulating natural and social phenomena which is derivative from previous successor StarLogo. Being written in Java enables NetLogo

Algorithm 5.1 Simulation Algorithm

```
1: loop
2:   make agents move randomly
3:   for every agent do
4:     if agent find other agent in the near then
5:       decide randomly who is the buyer b and seller s
6:       buyer asks other agents to make partial decision trust based on LG
7:       for every queried agent a do
8:         let ρ_as be opinion of a towards s
9:         if ρ_as/(1−ρ_as) > LG then
10:          a approves proposed transaction
11:        else
12:          a refuses proposed transaction
13:        end if
14:      end for
15:      if total agents who agree ≥ vote threshold then
16:        transaction proceeds !
17:        let s's fitness increases by predefined incentives value
18:        s gives the outcome randomly from his predefined space of out-
           come
19:        b rates the outcome and broadcast the rating to others
20:        for every agent a who receives the rating do
21:          agent a updated his opinion about s
22:        end for
23:      else
24:        transaction does not proceed
25:      end if
26:    end if
27:    let the fitness of s decreases by predefined amount of energy used in en-
       counter
28:    if fitness ≤ 0 then
29:      agent dies
30:    end if
31:  end for
32: end loop
```

to be run on any platform that Java supports. Furthermore, writing simulation program in NetLogo is relatively easy, since social environment *a'la* NetLogo is ready to be used.

NetLogo incorporates a function so-called *Behavior Space*. It is an integrated tools to be used in performing set of experiments on a model. It runs the model several times using predefined space of parameters settings which enables one to explore various behavior of the model from the combination of the parameters settings. This function will be used to conduct parameter sweeping on the model presented in the next section to see various behavior of the system.

In NetLogo, agents are modeled as moving turtles on a two dimensional area. The random movement of the turtles on the area allows the simulator to imitate random encounters. Two agents coincidentally in the near of each other are subjected for encounter of transaction. Algorithm 5.1 describes the simulation algorithm. In addition, the complete source code of the simulator program is documented in the appendix B.

5.2.1 Simulator Parameters

Figure 5.1 pictures user interface of the simulator. There are sliders, switches, and buttons allowing the user to set specific parameters in the simulation. It also has set of graphs displaying the dynamic of variables observed during simulation.

As previously discussed, the main parameters in the simulation are:

1. Initial distribution of ρ represented in $[0, 1]$. This parameter is set to determine the level of trust at the genesis of community.

2. LG ratio, real $[0, 10]$. The constant LG - Loss to Gain ratio values which is arbitrarily set to determine the transaction profile in the community.

Initial distribution of ρ and LG can be arbitrarily set on specific values or can be set randomly using switches below the sliders to imitate random initial level of trust or random transaction profile in each transaction.

In connection with the simulation environment itself, there are other parameters that can be set using sliders and switches:

1. Vote threshold real $[0, 100]$. Threshold value used in the voting to represent majority of agents.

2. Number of agents in each population of agents.

3. Natural selection related parameters: Initial fitness of each agent, the amount of energy used in every encounter, and the energy agent receives when agent is trusted to do transaction

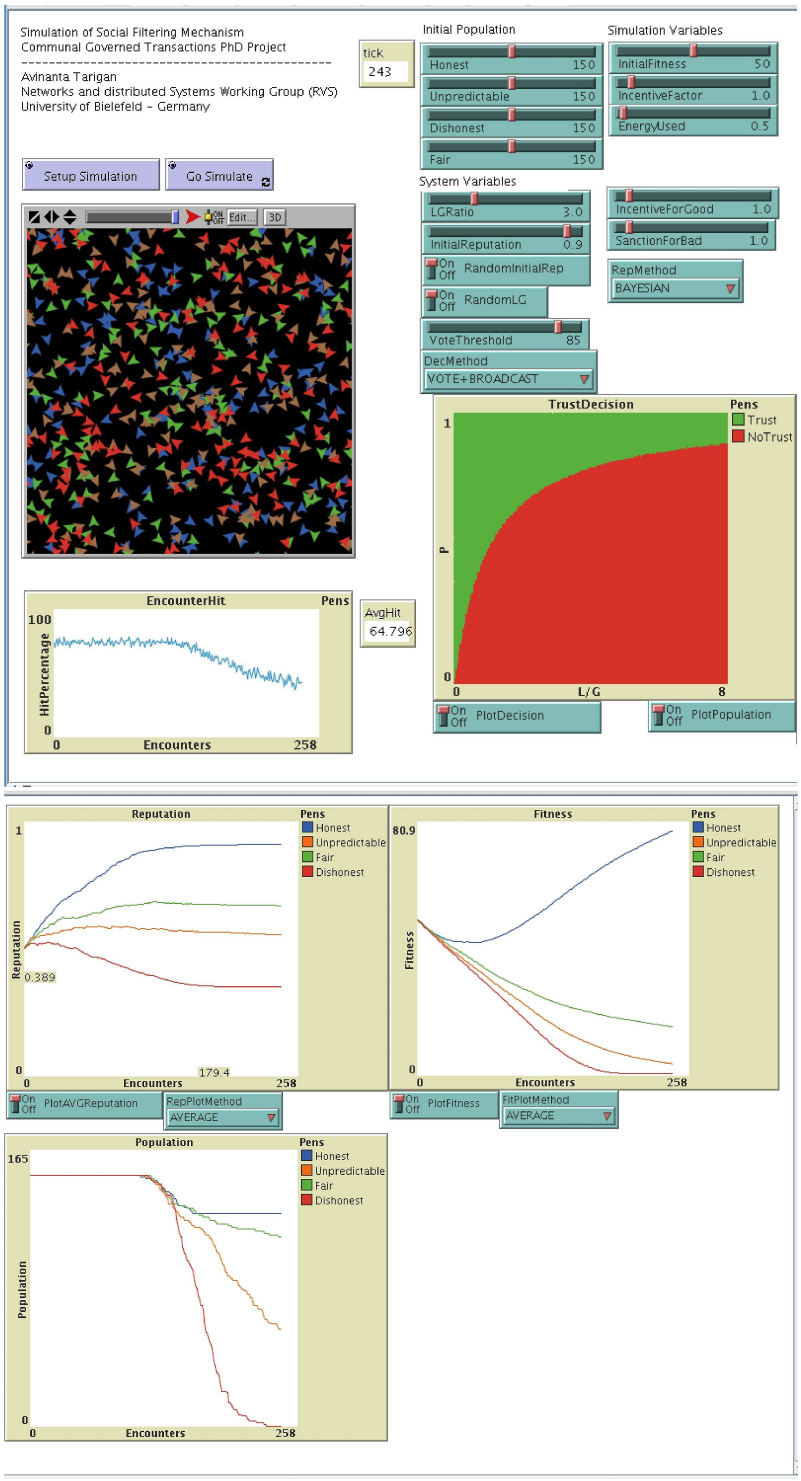


Figure 5.1: Simulator User Interface showing simulation on the collective authorization using random initial ρ and random LG in every encounter

4. Ratings parameter, incentive for good (value that is added to *sum_good*) and sanction for bad (value that is added to *sum_bad*) used in the Beta reputation. Note that this parameter is only used for trial experiments.

5. Threshold value to determine whether an outcome is good (1) or bad (0).

The simulator is programmed that it can simulate two different algorithm on the same set of data for comparison purposes. These algorithms are:

1. The '*Vote+Broadcast*' scheme which is the algorithm of collective authorization. Agents vote for community decision trust and the buyer broadcast the rating of transaction based on the outcome of transaction.

2. The '*Recommender*' scheme which is common recommendation algorithm. The buyer makes decision trust alone using Coleman's trust decision model (eq 4.1 on page 60) that is based on LG and reliability trust of seller derived from all recommendations.

In addition, two rating or recommendation computation method are incorporated in the simulator, namely, **Average** and **Beta Reputation**. These two methods are used to conclude reliability trust.

5.2.2 Display Graphs

In order to observe important variable change, three main displays of graphs are incorporated:

1. Reputation graph shows change of average value of distribution of ρ (reputation) towards each behavior types.

2. Fitness graph shows the change of average value of the fitness of each behavior types

3. Population graph shows the change of populations on each behavior types. This display is the most important graph which illustrates the natural selection process during simulation.

Reputation and fitness graphs display the variable: average value from all agents, or the values from one of randomly selected agents. Additionally, the encounter hit graph displays percentage of agents that are subjected for encounter on each movement. Next to it, trust decision monitor graph is used to monitor trust decision conducted by every agent given parameters ρ and LG. The red area shows "No Trust" and green area shows "Trust".

	Recommendation	Vote+Broadcast
Average	5096	251
Beta Reputation	6046	248

Table 5.2: Average Generation of Encounters until the system reaches a state where trust is assumed to be sustainable (Population of dishonests are very low but population of honest are still preserved)

5.3 Running Simulation

5.3.1 Parameter Setting

Preliminary runs are conducted to observe behaviors of the system under different set of parameter values. Based on the result, an optimal values of simulator parameters are gathered and used thorough experiments, namely:

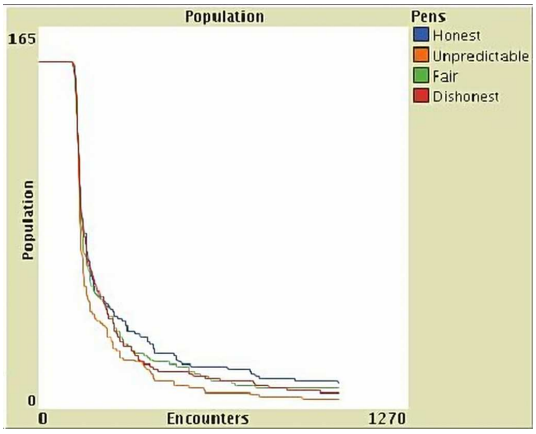
- Community is composed of 600 agents in which there are 150 agents from each behavior population. This composition of population along with the fixed size of two dimensional area on which agents move establishes approximately 64-75 % encounters per-random-movement.
- Initial fitness of every agent is set to 50
- Energy used which decreases fitness in every transaction is set to 0.5, whilst the incentive value which increases fitness is set to 1.
- Incentive for good and Sanction for bad are set to 1 according to the Beta reputation. Afterall, they are used only on trial runs.

These values are concluded after several trial runs of simulation in which the values of initial distribution of ρ and LG ratio are set to random.

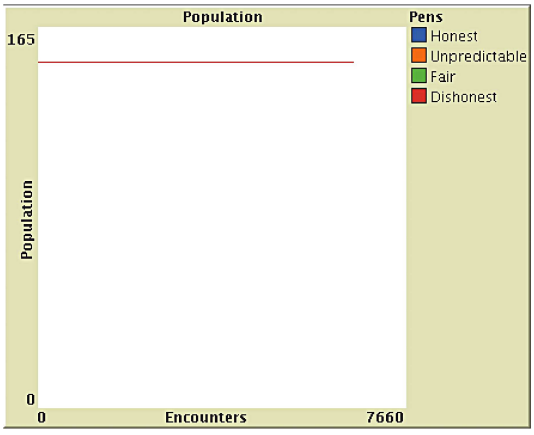
The next step is to run 100 simulation in which simulator parameters are set according to the values explained above, and system parameters - initial distribution of ρ and LG ratio, are set to random. Table 5.2 presents average generation of encounters in which system reaches a state where population of the dishonests are very low and the population of honest are still preserved. These values are used during experiments to determine the state at which the value composition of populations are observed.

5.3.2 Observed Phenomenon

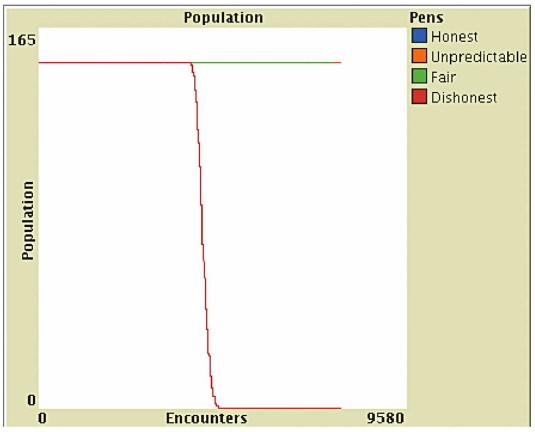
During simulation, there are conditions and trends that are observed in the evolution of population of agents. The following are classification those conditions, namely:



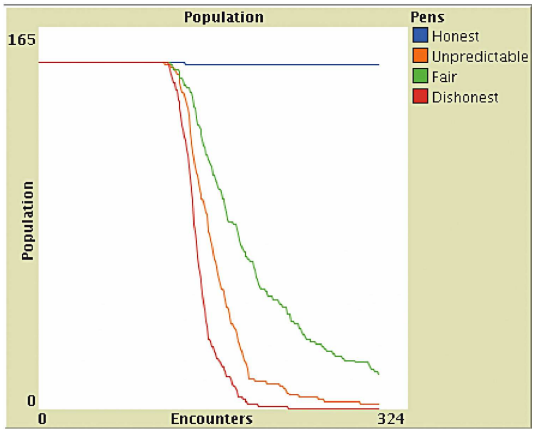
(a) Condition 1. All Agents are Extinct



(b) Condition 2. All Agents are Preserved



(c) Condition 3. Dishonests are Vanished, Others are Preserved



(d) Condition 4. Honests are Preserved. Dishonests are Extinct. Others might be Extinct.

Figure 5.2: The Four Conditions Observed During Simulation. These are the example of graphs showing the four conditions of populations of agents applying collective authorization (subfigures a,d) as well as recommender system (subfigures c,d)

Condition 1 Agents from any behavior type are extinct. At the end of simulation session, the typical composition of populations are: Honests<10 %, Dishonests<10%, Fairs<10%, Unpredictables<10%. Figure 5.2(a) shows the example of the graph showing how populations of each behavior type are decreased. This condition represents insufficient number of authorized transactions allowing agents to receive incentive in order to gain the fitness. That is, the level initial trust among agents as community is formed is insufficient to cope with the high risk transactions.

Condition 2 Agents from any behavior type are extant. Figure 5.2(b) pictures the example of population graphs showing that all agents are preserved even at the state where number of encounters generation is considered to be too high. In this context, the algorithm is failed to filter bad behaving agents.

Condition 3 Honest, Fair, and Unpredictable agents are extant, but Dishonest are extinct. The typical composition of population in this state are: Honests>95%, Dishonests<5%, Fairs 80-100%, Unpredictables 80-100%. Figure 5.2(c) pictures the example of population graphs showing this condition. The rests of the runs after snapshot indicates stabilizes trends of population of the Honest, Fairs, and Unpredictables.

Condition 4 Honests are extant, dishonests are extinct, and both Fair and Unpredictables tend to be extinct. The typical composition of population is Honest>95%, Dishonest<10%, Fair<30%, Unpredictable<30%. Figure 5.2(d) pictures the example of population graphs showing this condition.

These four conditions measure how successful collective authorization algorithm in filtering bad behaving agents and at the same time providing space for good behaving agents to continue to trade. They are used in determining result of experiments described in the next section.

5.4 Experiments

5.4.1 Parameters Sweeping

NetLogo has a feature so-called Behavior Space which allows one to explore the model's "space" of possible behaviors and determine which combinations of parameter cause the behaviors of interest. This is done by varying parameters according to defined set of values.

Particularly, those parameters are initial distribution of ρ - initial trust level of community, and LG ratio - the type of goods or service being exchanged in the community. The objective is to understand the possible effects of every combination

of both parameters in the system. The result is intended to be used as consideration in implementing the framework in the future.

In this experiments, parameters sweeping is conducted by varying initial distribution of ρ using interval 0.05, starting from 0.05 - representing low initial trust, to 1.0 - representing high initial trust. The LG is varied starting from 0.3 - representing low risk and high potential benefit transactions, to 10.0 - representing high risk transactions, using interval 0.3. Whilst other simulator parameters stay constant. These combinations requires 500 simulation runs. In addition, each 500-runs is repeated 5 times on different set of random encounter generations to see consistency of the results.

The experiments are also done for simple average and Beta reputation in order to observe the influence of reputation computation method used in the algorithm. Furthermore, experiment applying the same technique is also conducted on recommender system for comparison purposes.

Being written in Java gives NetLogo its multiplatform capability. However, it has drawback on the performance. Running one run of simulation takes approximately 1-2 hours on Pentium IV 2.3 Ghz 1Gb RAM machine running Linux. Therefore, experiments are splited up to be run on several machines. In order to make sure that the machines delivers the same set of random process, each runs uses the same random-seed. It guarantees that the pseudo-random generators on each simulation begins with the same starting state and runs the same random behavior.

5.4.2 Observation

Observation is focused on populations of agents from each behavior type. The interest is laid on the composition of population at or after the r^{th} encounter. The value of r is determined using approximation of the result from preliminary run presented on table 5.2 on page 76. Here, the r^{th} encounter is considered to be the step where the algorithm should "successfully vanish" the dishonests.

In simulation on Recommender system, the snapshot is taken at the $r = 7000$, and at $r = 350$ for Vote+Broadcast algorithm. Data on each snapshot is categorized according to the four conditions previously described. Furthermore, the step of encounter generation at which dishonests are extinct on each run is also noted to understand the performance of the system.

5.4.3 Presenting Results

Overall results of experiment are drawn on graphs presented in figures 5.3 - 5.10. Graphs on figure 5.3, 5.4, 5.7, and 5.8 present classification of each population composition snapshoted from the r^{th} encounter in parameter sweeping. The red dots indicates condition 1 where all agents are extinct. The yellow indicates condition

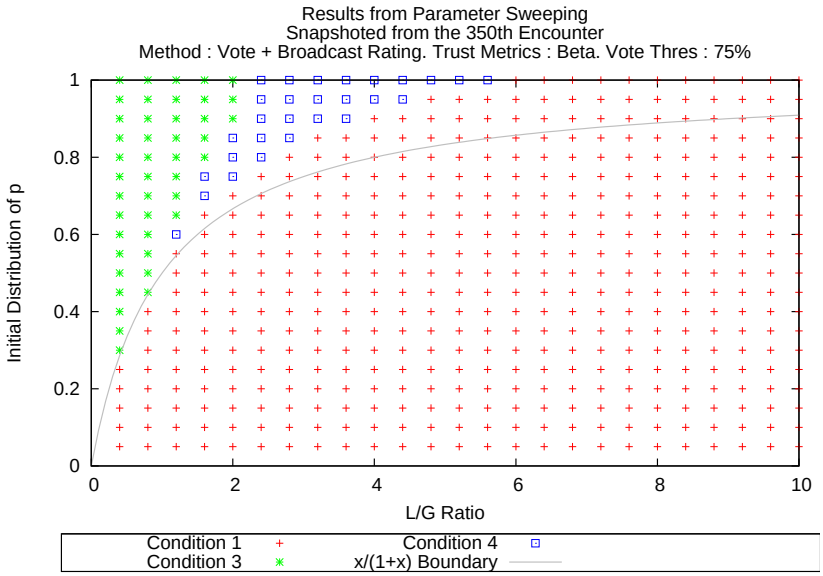


Figure 5.3: Results from Parameter Sweeping on Collective Authorization Using Beta Reputation. Each point represents snapshot of population from the 350^{th} encounters. Each point is colored according to the four conditions. Red, Yellow, Green, and Blue points represent in respective manner, condition 1, 2, 3, and 4.

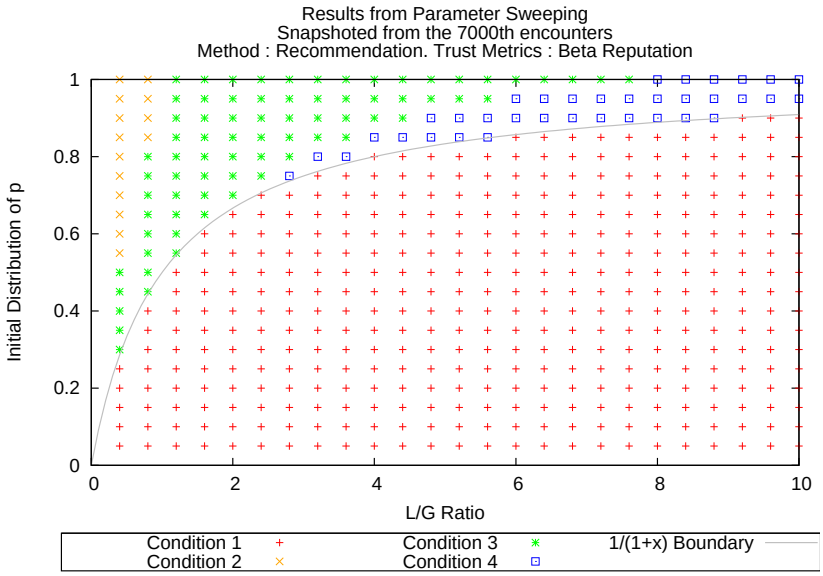


Figure 5.4: Results from Parameter Sweeping on Recommender System Using Beta Reputation. Each point represents snapshot of population from the 7000^{th} encounters. Each point is colored according to the four conditions. Red, Yellow, Green, and Blue points represent in respective manner, condition 1, 2, 3, and 4.

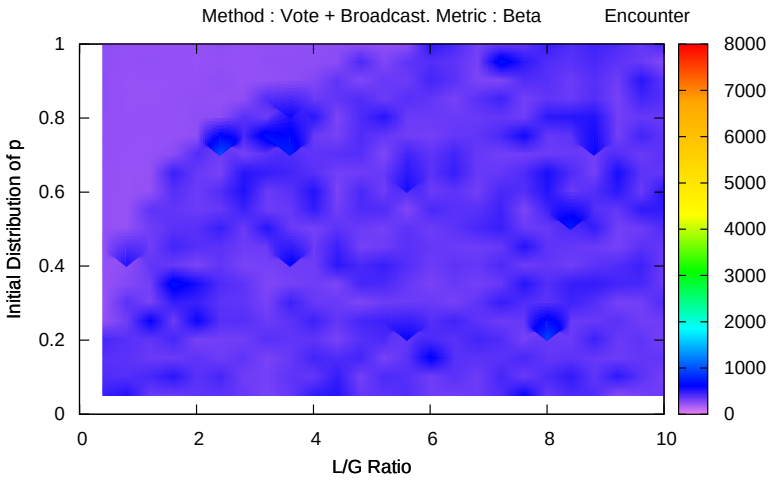


Figure 5.5: Performance of Collective Authorization using Beta Reputation. The Graphs shows time needed by agents applying this algorithm to vanish the Dishonests. One point in the graph shows how many encounter is required until dishonest become extinct.

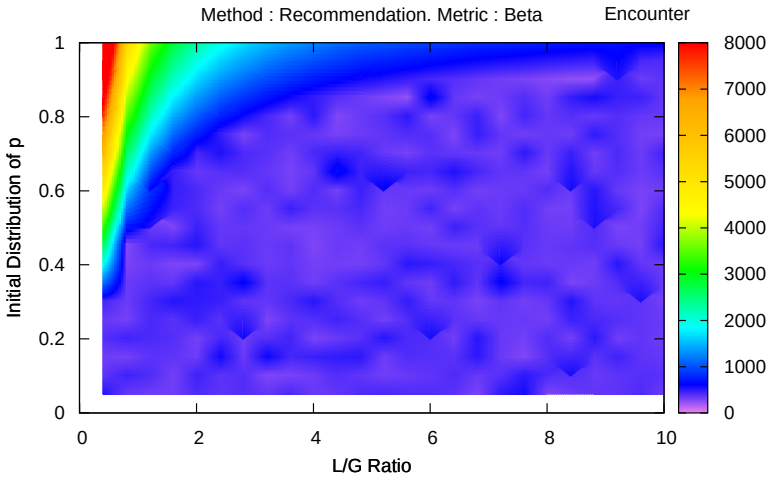


Figure 5.6: Performance of Recommender System using Beta Reputation. The Graphs shows time needed by agents applying this algorithm to vanish the Dishonests. One point in the graph shows how many encounter is required until dishonest become extinct.

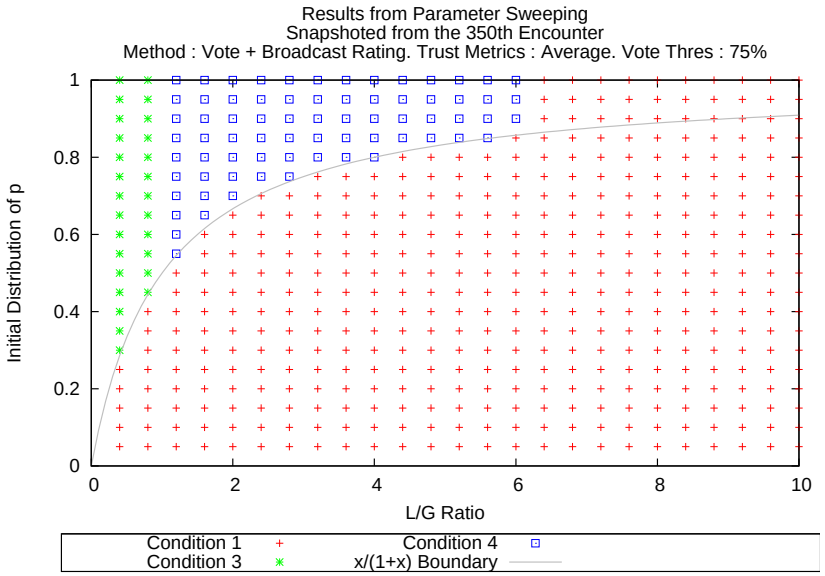


Figure 5.7: Results from Parameter Sweeping on Collective Authorization Using Simple Average. Each point represented snapshot of population from the 350^{th} encounters. Each point is colored according to the four conditions. Red, Yellow, Green, and Blue points represent in respective manner, condition 1, 2, 3, and 4.

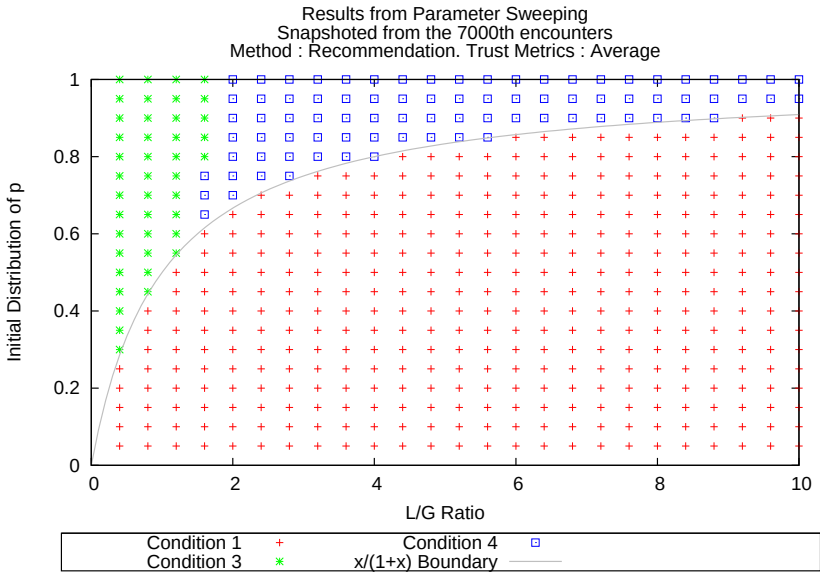


Figure 5.8: Results from Parameter Sweeping on Recommender System Using Simple Average. Each point represented snapshot of population from the 7000^{th} encounters. Each point is colored according to the four conditions. Red, Yellow, Green, and Blue points represent in respective manner, condition 1, 2, 3, and 4.

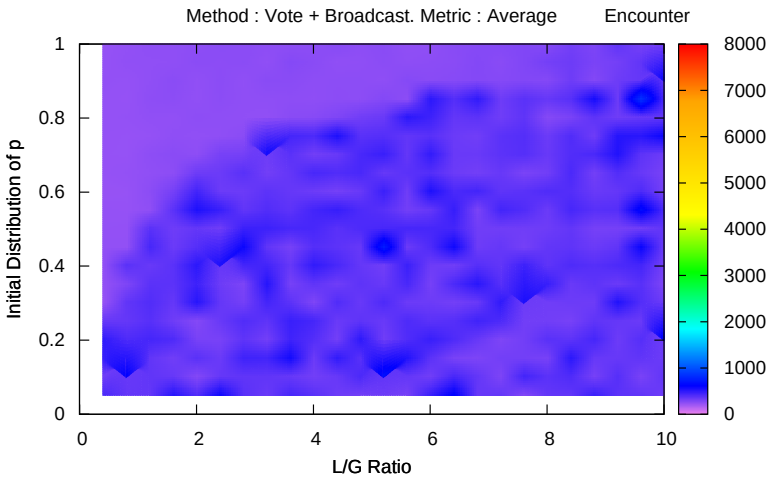


Figure 5.9: Performance of Collective Authorization using Simple Average. The Graphs shows time needed by agents applying this algorithm to vanish the Dishonests. One point in the graph shows how many encounter is required until dishonest become extinct.

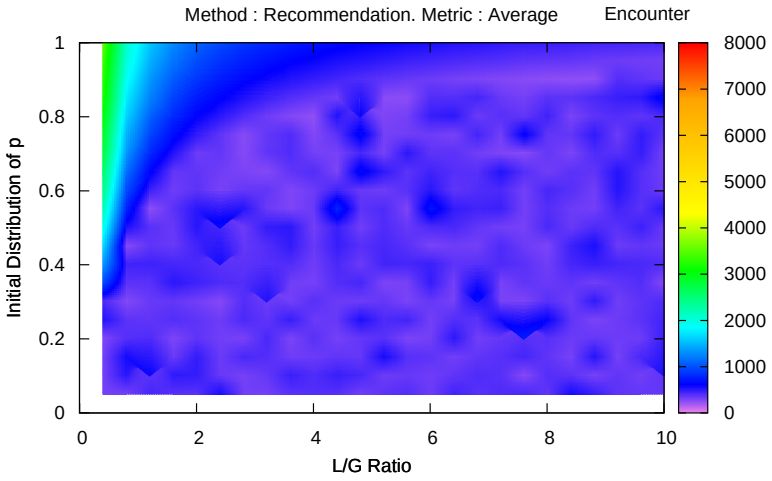


Figure 5.10: Performance of Recommender System using Simple Average. The Graphs shows time needed by agents applying this algorithm to vanish the Dishonests. One point in the graph shows how many encounter is required until dishonest become extinct.

2 where agents are preserved. Green, and blue dots indicate condition 3, and 4 respectively. Additionally, a curve *initial distribution* $\rho = \frac{LG}{1+LG}$ is drawn as helper line in interpretation of the graph. This curve separates area of go and no-go in the rational decision trust. Above the curves are the area of go for decision trust and under the curve is the area of no-go. The line itself defines the indifferent decision trust where $\frac{\rho}{1-\rho} = LG$.

Figure 5.5, 5.6, 5.9, and 5.10 show the performance of algorithms at each parameter combination. Performance is assumed to be amount of time needed by algorithm to vanish the dishonests. Thus, each dot in the graph is the measure of encounter generation at which dishonest become extinct. In the graph, the value of encounter generation is represented by palette of colors. Additionally, all data in each graph are interpolated from which gradual colors in the graphs are generated. It advances visualization of the graph for the shake of esthetic.

5.5 Discussion

The result presented in figure 5.3, 5.4, 5.7, and 5.8 show that at any combination of initial distribution of ρ and LG, which are under the helper curve, brings the system into the state where condition 1 is reached. It shows that the level of initial trust is insufficient to cope with the risk of transactions. Having this condition, agents do not have any chance to transact and gain their reputation. That explains why at the end all agents are extinct.

In this respect, performance graphs show that the time for algorithm to vanish the Dishonests varies approximately 50 to 1000 encounters generations. In this area, no pattern exhibited by the graph due to the way the simulator generates encounter which is randomized movement of turtles on an area. The less turtles exists, the less chance that turtles meet each other. This circumstance applies to all experiments which run Recommender system and Vote+Broadcast scheme using any trust metric.

On the other hand, any combination parameter values above the curve affects the system in slightly different ways. The use of Recommender system, for instance, brings system to the states where condition 2, 3, and 4 are reached. Figure 5.4 and 5.8 show that the system reaches the state where condition 4 occur as the value of LG ratio gets higher. Conversely, when LG ratio gets lower the system reaches the state of condition 3. In other circumstances, the system exhibits condition 2 when the value of LG is small and initial ρ is high. In fact, it shows the common aspect in rational decision trust, that the more risky the transaction is, the more trust one needs.

One might notice in Recommender system, that simple average slightly outperform Beta reputation. This is confirmed by Liang and Shi [Liang and Shi, 2005] who performed evaluation of several trust metrics, but not by Schlosser et. al. [Schlosser

et al., 2004] who did the evaluation with different approach. The reason is that Recommender system incorporated in this simulation counts the whole recommendations. Different models of Recommender System count opinions only from subset of agents.

Moreover, Liang and Schlosser measured performance of trust metrics by observing dynamics of the reputation, not the emergent property of the system which is done in this simulation. Nevertheless, since this thesis does not concern about the trust metrics performance in depth, the performance issues in Recommender system affected by different trust metrics are not discussed here.

The interesting part of the result is that performance of Vote+Broadcast scheme surpasses Recommender system. The time for the Vote+Broadcasting scheme to exterminate the Dishonests is lots faster than Recommender system. Moreover, the Recommender system exhibits linear degradation of performance as the value of initial ρ gets higher and LG gets lower. On the contrary, the results exhibited by Vote+Broadcast scheme demonstrates that distribution of the performance are steady for any value of initial ρ and LG.

Nevertheless, the simulation shows limitation of Vote+Broadcast scheme. The system reaches the state of condition 2 as LG ratio gets higher although it is above the helper curve. In fact, using Beta reputation scheme, the maximum LG ratio which system can handle is approximately 5.6. This value is taken in condition that initial trust level of the community is the highest. On the other hand, simulations that use of simple average show that the maximum value of LG than system can handle is 6.

Conclusively, graphs presented here show approximate operating range of collective authorization which can be used as considerations in the future works. One should avoid the area that exhibited red or yellow dots and try to stay on blue or green. Consider that a group of people want to set up MP3 trading community in which transaction risk is assumed to have $LG = 1$. They should be sure that the initial trust level among them is at least 0.6.

5.6 Summary

This chapter presents the result of simulation on collective authorization. It demonstrates how social control works in that the system is able to detain Dishonests more effectively rather then normal Recommender system. Nevertheless, the study shows that collective authorization scheme has limitation. It can be used by the community with relative low-risk trading profile. This chapter also reveals approximation of operating range of the collective authorization. It is regarded to be the considerations in the future implementation of proposed framework.

Chapter 6

The Design of the Protocol

This chapter investigates the requirements and issues in the implementation of the transaction algorithm presented in previous chapters and delivers best-practice design of the transaction protocol. The first section draws the scenario and reveals issues which might arise during implementation. The discussion leads to the design of a set of protocols utilizing distributed public-key crytography to devise mutual authentication problem. The subsequent section describes the protocols concerned with infrastructure of the transaction: community establishment, key management, as well as the memberships. The third section specifies the transaction protocol which is the extention of the transaction mechanism proposed in previous chapters in which some arbitrary modifications are implemented for the shake of feasibility in practice. This chapter ends with the summary and discussion.

6.1 The Scenario, Issues, and Intended Solution

6.1.1 The Scenario

Consider a P2P trading community C consists of n agents[1] who conduct trading among them without the existence authority of server to facilitate transactions. Community grows when new member joins in and shrinks when existing member leaves or being convicted by community for some reason. A public network facilitates communication among agents where address resolution is already established. Here, agents are assumed to be reliable but not always reachable. That is, every message transmitted through the network is guaranteed to be received by destined agent. These assumptions is arbitrarily taken to allow higher level of abstraction of the protocol.

The developed protocol assumes that community seed is already established by a

[1]The notion of agent is used instead of peer in order to have similarity in explanation and to avoid confusion

mutually trusted agents so-called *the pioneers*, through discussions in mailing-lists, blogs network, or even from offline meetings. In this stage, the pioneers already agree on set of initial institutional facts establishing membership and community money. They also agree on the distribution of community money to the agents.

In the establishment of community, agents perform bootstrapping protocol explained in section 6.2.4. Since the community is semi-closed system, new member can join in with two possibilities: either by being recommended by one of the members, or by selling his resources to the one of the member consequently receiving the corresponding institutional-money and membership. The latter enforces the new member to invest his goods or services to the community and thus encourages him to continue to trade within the community. The protocol that deals with enrolling new member is described in section 6.2.5.

In particular case where majority of agents agree that an agent should be convicted, or when an agent resigns from the community, agents perform expelling member protocol illustrated in section 6.2.6.

Each agent in the community has roles to trade as buyer or as seller as well as to perform his obligation in collaborative actions to complete transactions. Transactions are conducted by performing transaction protocol explained in section 6.3.

6.1.2 Issues and Intended Solutions

Since agents utilizes public network the first issue that arises is authentication. How agents can verify whether received message is genuine. Digital signature has been known to devise such problem by providing a mechanism to authenticate the source of message utilizing public-key cryptography. However, digital signature requires key administration by which property of so-called key-trust, the trust in the context of network security, is established. One believes that a key belongs to someone when the key administrator, the Certification Authority, certifies the binding between the owner and the key which is known as digital certificate. Here, CA establishes the hierarchy trust model which simplifies the introduction of members' public-keys. In contrast to this, the aim of the design is to avoid the use of such service. This leaves the key-trust and key-management issues open.

Furthermore, there should be a way to justify that an assertion is collectively accepted as institutional-fact. The mechanism should prevent a state where agents deny those institutional-facts, as it might lead to the dispute or distrust. This property, entitled with "*Multiparty Non-Repudiation*" (MNR), can be established, for example, by generating agent's digital signature of an assertion to signify agent's acceptance on particular assertion. Here, signatures of all agents justify that an assertion is institutional fact. However, this solution is impractical since attaching all signatures would require more space and verification would require computation time as community expands.

The protocol should also prevent double-spending of institutional-money. In particular, the protocol should devise how agents justify that an assertion is no longer accepted as institutional fact. The thinkable solution is to maintain communally signed list of eliminated institutional facts (LEI). In analogy to PKI, this list is called Certificates Revocation List (CRL) maintained by CA. Still, this solution is also impractical because content synchronization as well as mutual signing would be too costly to be collectively performed by agents. Moreover, it limits the performance of the system since a number of transactions can be completed on a window of collaboration in altering and signing the LEI.

Despite the non-existence of TTP implies that solution could not be as simple as when TTP facilitates the transaction, the protocol design should maintain practicability in its implementation. Therefore, the following illustration is proposed to solve those issues.

- Distributed cryptography offers a solution to achieve MNR. Agents generate community-keys of which the private-key component is splitted and shared among agents. Agents use shared community private-key to generate partial-signature of the assertion. All partial-signatures can be computed into community-signature of the particular assertion. Not only that it simplifies signature verification procedure, but this method also distributes the authority of the community key among agents. Using this scheme, every agent gets an equal role in the community signature. The detailed discussion of distributed cryptography and the achievement of MNR is discussed in the next section.

- Using distributed cryptography, community signature on an assertion signifies that the assertion is institutional fact. Since MNR is achieved, no agent can deny that the assertion is institutional fact. This implies that institutional-money and institutional-memberships are the count-as assertions which have community signature attached.

- To devise key-trust problem, public-key of an agent is incorporated as an attribute in his institutional-membership. Since community signature is attached on it, institutional-memberships represents certified binding between agent-id with his public-key. In analogy to key-trust established in PKI, agents perform the function of CA in collective manner. Marchesini and Smith [Marchesini and Smith, 2002] introduced this as "Virtual Hierarchies" in distributing key-trust on several intermediate CAs.

- In order to prevent double spending, an institutional-money should has serial number incremented during each transaction. Using this scheme, who stated as owner in an institutional-money that has highest serial number is the current

owner of that particular money. Since distributed cryptography requires all agents to perform community signature generation, double-spending attempt can be countered by the owner of the institutional-money.

- As previously mentioned, the protocol assumes that community seed is already established by a mutually trusted agents so-called *the pioneers* who are able to authenticate each other using their public-keys. Agents can either use PGP, Key Exchange algorithm [Blum, 1983], or any other media to exchange their public-keys. The pioneers, denoted with Ψ where $\Psi \subseteq C$, is assumed never leave community. The existence of pioneers are intended to solve the problem of trust in the establishment of community as well as the community private-key redistribution problem. That is, they are the anchor of both key-trust and the real trust in the community. Despite that all agents have equality to play any role in the transaction, yet the assumption of the pioneers should sociologically make sense.

6.2 Bootstrapping, the Keys, and Memberships

6.2.1 Protocol Basics

Public-key cryptography or asymmetric cryptography is used heavily to provide authentication as well as non-repudiation. Therefore some notations related to it have to defined before its use. Protocol descriptions in this chapter use a_i (with index) for denotating an agent in community C that consists of n agents $\{a_1 ... a_n\}$. In some parts of description, a_b and a_s denotate the transacting agents, the buyer and the seller. Objects related to an agent are identified directly with the index of the agent. For example k_i denotes public-key of agent a_i.

Notations for Cryptographic Operations

Each agent $a_i \in C$ has private and public-key pairs denoted with k_i^{-1} and k_i, respectively. In order to achieve authentication of message msg, agent a_i the message using his private-key. This process produces agent a_i's digital-signature of msg denoted $sig_i msg$ which is the result of:

$$sig_i msg = encrypt\left(k_i^{-1}, hash(msg)\right)$$

where *hash* is any one-way-hash function that produces "finger-print" of message and *encrypt* is the encryption function of a cryptography algorithm. Let *decrypt* be decryption function of corresponding *encrypt*, signature $sig_i msg$ can be verified

only using corresponding public-key k_i:

$$sigverification(k_i, msg, sig_i msg) = \begin{cases} valid & if\ hash(msg) = decrypt(k_i, sig_i msg) \\ not\ valid & otherwise \end{cases}$$

Any message transmitted between agents should be signed by the sender(s) by attaching its / their signature(s) in the message as well as other information which is essential in the authentication process. In order to simplify the notation, message msg signed by agent a_i is denoted with $\{msg\}_i$ where

$$\{msg\}_i = \langle msg, \{\hat{p}_i\}_C, sig_i msg \rangle$$

$\{\hat{p}_i\}_C$ is institutional-membership of agent a_i containing a_i's public-key k_i. Public-key contained in the institutional-membership is important for signature verification. One can verify that signed message msg is genuine from agent a_i by comparing the hash of received message msg with the result of the decryption of $sig_i msg$ using a_i's public-key k_i contained in $\{\hat{p}_i\}_C$. The detailed explanations of institutional-membership and the community signature are discussed in the next section.

There are events in the system, where two or more agents agree on a message. Together, they sign that message and attach their signatures as well as institutional-memberships. Consider agents a_b and a_s agree on message msg and together they sign that message denoted with $\{msg\}_{b,s}$ such that

$$\{msg\}_{b,s} = \langle msg, \{\{\hat{p}_b\}_C, \{\hat{p}_s\}_C\}, \{sig_b msg, sig_s msg\} \rangle$$

Using the same way to express the signaturing, set of agents $\{a_i ... a_j\}$ sign message msg is denoted with $\{msg\}_{i...j}$ such that

$$\{msg\}_{i...j} = \left\langle msg, \left\{\{\hat{p}_i\}_C ... \{\hat{p}_j\}_C\right\}, \{sig_i msg ... sig_j msg\} \right\rangle$$

Notations for Communicating Agents

This chapter uses two standard communication operators: send denoted with "$\mapsto$" and broadcast denoted with "$\Leftrightarrow$". Agent a_b sends signed message msg to agent a_s is expressed with:

$$a_b \mapsto a_s : \{msg\}_b$$

Agent a_b sends signed message msg to set of agent $\{a_i ... a_j\}$ is expressed with:

$$a_b \mapsto \{a_i ... a_j\} : \{msg\}_b$$

Using assumption that the first agent collects the signatures and sends the message to the receiver, a set of agents $\{a_i ... a_j\}$ sign message msg and send it to agent a_s is expressed with:

$$\{a_i ... a_j\} \mapsto a_s : \{msg\}_{i...j}$$

Using the same way, a set of agents $\{a_i ... a_j\}$ sign message msg and send it to set of agents $\{a_k ... a_l\}$ is expressed with

$$\{a_i ... a_j\} \mapsto \{a_k ... a_l\} : \{msg\}_{i...j}$$

Agent a_b signs and broadcasts message msg to the community C is expressed with:

$$a_b \diamondsuit C : \{msg\}_b$$

Using the same assumption, that the first agent collects the signatures and sends the message to the receivers, a set of agents $\{a_b, a_s\}$ sign message msg and broadcast it to the community C is expressed with

$$\{a_b, a_s\} \diamondsuit C : \{msg\}_{b,s}$$

The reason that broadcast operator is not defined as action of multiple sending is that the availability multicast channel allowing agent to take advantage by sending the message only once through that multicast channel instead of having repeatly send the message to every agent in the community.

6.2.2 The Using of Distributed Cryptography

Some Background

Since it was introduced by Shamir [Shamir, 1979], distributed cryptography has been receiving great interests. Instead of one-sender-one-receiver performed in traditional cryptography, distributed cryptography considers many to one or one to many sender/receiver in the system [Desmedt, 1998]. The idea is to distribute the authority of the secret or private-key to group of stakeholder. The cryptography operation in which the shared private-key is required must be performed by those stakeholder. In this scheme, entitled with n-out-of-n, all stakeholder are responsible to any result of cryptography operations (encryption, decryption, signature generation) involving the shared private-key.

In respect to the proposed protocol, the generation of community digital-signature of an assertion requires all agents to be involved using his part of shared community private-key. Here, community signature of an assertion signifies that assertion is accepted by all agents. Since no community signature can be generated

without involvement of all agents, the system achieves MultiParty Non-Repudiation (MNR).

In the development, the so-called threshold cryptography was developed to allow m-out-of-n stakeholder to perform the cryptography operation. Here, the presence of m stakeholder is sufficient to perform intended cryptography operation. Mainly, this scheme is used to achieve fault tolerant in distributed system. Transaction protocol proposed in this thesis utilizes n-out-of-n distributed cryptography. Not only due to simplicity of the protocol, but the property of the key enables agents to prevent double spending attempt.

Notations of Community Signaturing

Community C has private and public-key pairs, denoted with $k_C^{-1}$and k_C respectively, in which community private-key k_C^{-1} is splitted in a way that every agent $a_i \in C$ possesses partial community private-key $k_{C:i}^{-1}$. The algorithm allows an agent to generate partial community signature of message msg denoted with $psig_{C:i}msg$, such that

$$psig_{C:i}msg = encrypt\left(k_{C:i}^{-1}, hash(msg)\right)$$

The complete community signature $sig_C msg$ of message msg is constructed from all partial signatures such that,

$$sig_C m = sigconstruct\left(\bigcup_{a_i \in C} \{psig_{C:i}msg\}\right)$$

where *sigconstruct* is a function which takes all partial signatures as input and produces complete community signature. Here, the system uses standard signature verification process in which community public-key k_C is used. Here, community signature $sig_C msg$ on message msg is valid when $hash(msg)$ is equal to $decrypt(k_C, sig_C msg)$.

In order to capture distributed cryptography operation, basic notations explained earlier in this section is extended. Agent a_i partially signs message msg using his part of shared community private-key $k_{C:i}^{-1}$ is denoted with $\{msg\}_{C:i}$ where

$$\{msg\}_{C:i} = \langle msg, psig_{C:i}msg\rangle$$

Consequently, a set of agents $\{a_i ... a_j\} \subset C$ sign message msg using their part of shared community private-key $k_{C:i}^{-1}...k_{C:j}^{-1}$ is denoted with $\{msg\}_{C:i...j}$ where

$$\{msg\}_{C:i...j} = \langle msg, \{psig_{C:i}msg...sig_{C:j}msg\}\rangle$$

RSA based Distributed Cryptography and the Property of Community-Key

This protocol utilizes n-out-of-n distributed RSA public-key scheme [Malkin et al., 1999, Boneh and Franklin, 2001] in which private-key component is splitted **in additive form** and distributed among signature participants. This system allows agents to perform simple signing and simple signature verification as well. Furthermore, the protocol takes advantage from simple partial-keys management in order to establish community key regeneration in the event of membership's change.

Let $k_C^{-1} = \langle d_C, N_C \rangle$ and $k_C = \langle e_C, N_C \rangle$ be community private-key and public-key pairs in RSA system where d_C is private-key component of k_C^{-1}. Pick n random integers $d_{C:i}$ satisfying that

$$d_C = \sum_{i=1}^{n} d_{C:i}$$

Every $d_{C:i}$ is given under authority by agent a_i. Here, the structure of shared community private-key held by agent a_i is denoted with $k_{C:i}^{-1}$ where $k_{C:i}^{-1} = \langle d_{C:i}, N_C \rangle$. This additive splitting method allows agent i to encrypt integer $z \in \mathbb{Z}$ using his partial community private-key $k_{C:i}^{-1}$:

$$encrypt\left(k_{C:i}^{-1}, z\right) = z^{d_{C:i}} \ \ (mod\ N_C)$$

where N_C is the modulus component of the community key. One might notice that the above operation is a standard RSA encryption operation in which the modulus operation keeps the $d_{C:i}$ secret. Thus, the complete encryption of z is constructed with

$$encrypt\left(k_C^{-1}, z\right) = \prod_{i=1}^{n} encrypt\left(k_{C:i}^{-1}, z\right) \ \ (mod\ N_C)$$

which is equal to

$$z^{d_C} \ = \ z^{d_{C:1}+\ldots+d_{C:n}} \ \ (mod\ N_C)$$

Let ζ be the result of encryption above, the decryption can be performed directly using community public-key component e_C such that

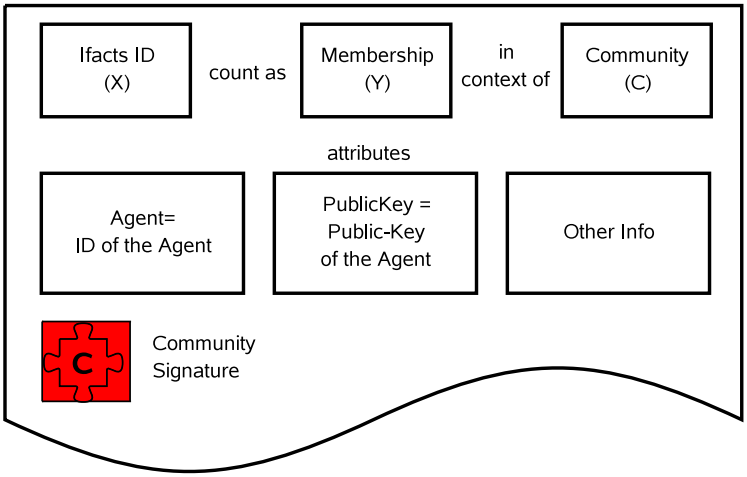
$$z = \zeta^{e_C} \ \ (mod\ N_C)$$

Combined with one-way-hash function, the distributed cryptography scheme can be used to generate partial-signatures as well as the complete community signature of a message msg where $z = hash(m)$. This partial signature scheme enables simple shared community-key management. The system works with any structure of splitting values as long as the sum of $d_{C:i}$ is equal to d_C.

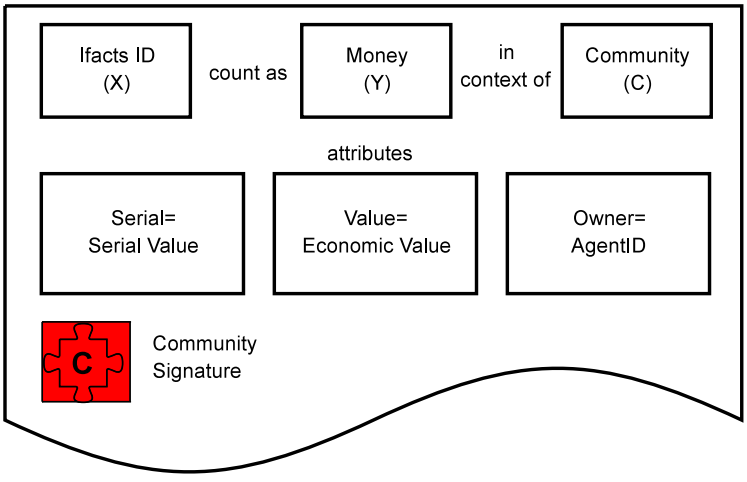
The fundamental problem in using such method is how to generate and to split

the community private-key d_C in which no agent or any entity whatsoever knows original value of d_C. Fortunately, Boneh et al. has developed method to generate RSA keys distributively such that the resulting private-key is splitted in additive form, and that no agents learns others' shared community private-key. Thus, this method is suitable to be used to generate community-key which is done only once when community is established.

6.2.3 Anatomy of Institutional-Money and Institutional-Memberships



(a) Institutional-Membership



(b) Institutional-Money

Figure 6.1: Anatomy of the Assertions establishing Institutional-Facts

In this framework, institutional-facts are arbitrary assertions which are accepted by the community. One recognizes that an assertion is institutional-fact by verifying community signature on the assertion using community public-key. There are two types of assertions established institutional-facts used in the framework, namely institutional-membership which signifies that an agent is member of the community, and institutional-money that functions as generally accepted medium of exchange within community. Figure 6.1 presents pictorial explanation of the two assertion types.

Essentially, the structure of count-as assertions in this chapter is similar to those explained in chapter 3 with slight modification to meet requirement of the transaction protocol. Assertion constructing institutional fact that establishes the membership of agent a_b (institutional-membership) is denoted with $\hat{p_b}$ where $\hat{p_b} = \langle id_{\hat{p_b}}, membership, C, \{agentid = a_b, publickey = k_b, otherinfo\}\rangle$. It contains identification of the agent valued in *agentid*, the public-key of the agent in *publickey*, and any other information regarding to the agent stated in *otherinfo*. If community accepts the assertion, then community signature on the assertion is attached. Therefore, institutional-membership created from assertion $\hat{p_b}$ is denoted with $\{\hat{p_b}\}_C$. Since community signature signifies the acceptance of all agents, the semantic of $\{\}_C$ is equal to the modal operator $\odot_C$ presented in chapter 3.

Assertion establishing institutional-money is denoted with $\tilde{p}$ where $\tilde{p} = \langle id_{\tilde{p}}, money, C, \{serial = s_{\tilde{p}}, value = val, owner = b\}\rangle$. The $id_{\tilde{p}}$ is unique number assignment of assertion in community C and *serial* attribute is the serial of to identify recentness of the assertion. It will be used to prevent double spending as well as to identify the last owner of the particular money. Furthermore, economical value of the money is stored in *value* attribute. The agent who owns it is stated in *owner* attribute.

If community accepts the assertion, then community signature on the assertion is attached. Therefore, institutional-money established from assertion $\tilde{p}$ is denoted with $\{\tilde{p}\}_C$.

In the real implementation, one might assign unique integer in the field *id* of assertion. Strictly speaking, the unique number valued in *id* is what should be named institutional-money or institutional-membership as it stated in Searle's formula "x counts as y in context of c". Since the framework proposed in this thesis is the result of engineering of those notions, the description does not take this matter into account and shall simply consider the whole assertion as institutional-money or institutional-membership.

6.2.4 Bootstrapping

Bootstrapping consists of stages which the pioneers perform at the establishment of community. As explained earlier, the assumption taken is that the pioneers are

mutually trusted agents know each other public-key and thus are capable to authenticate one another. Figure 6.2 illustrates bootstrapping stages discussed in the

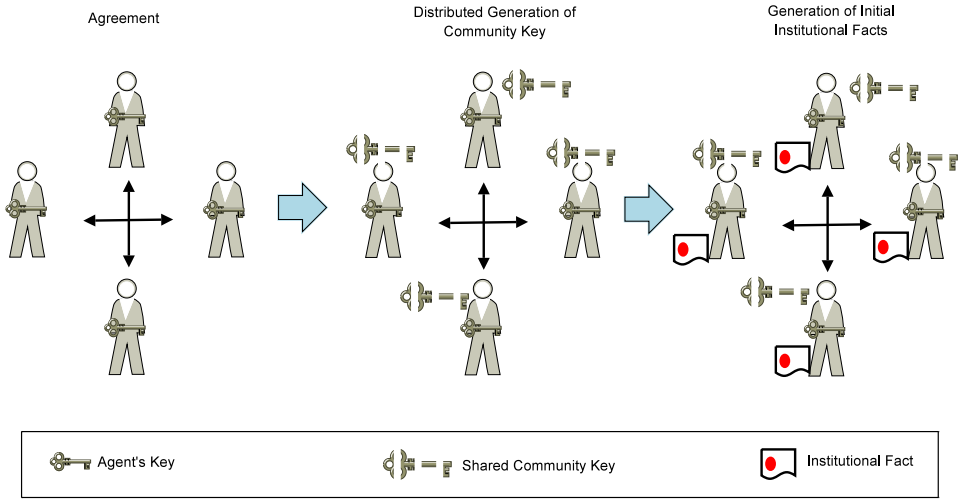


Figure 6.2: Community Bootstrapping

following descriptions:

Stage 1. Communal-key generation. Agents perform the Distributed RSA key Generation Protocol proposed by Boneh et. al. At the end of the protocol each agent $a_i \in C$ possess partial community private-key $k^{-1}_{C:i}$ and community public-key k_C. Using this protocol community private-key is splitted in additive form as explained in the previous section.

Stage 2. Generation of institutional-memberships. Agent performs the generation of institutional-membership for each agent. For all agent $a_i \in C$ broadcast membership proposal $\hat{p}_i = \langle id_{\hat{p}_i}, membership, C, \{agentid = a_i, publickey = k_i, otherinfo = addinfo\}\rangle$:

$$(\forall a_i \in C)\ a_i \overset{\nearrow}{\underset{\searrow}{\Leftrightarrow}} C\colon \{\{\hat{p}_i\}_{C:i}\}_i$$

Upon receiving the message, each agent $a_j \in C, a_j \neq a_i$ verifies the signature, the id, as well as the public-key stated in the proposal and gives his partial signature on the proposal to be broadcast to the rest of agents:

$$(\forall a_j \in C, a_j \neq a_i)\ a_j \overset{\nearrow}{\underset{\searrow}{\Leftrightarrow}} C\colon \left\{\{\hat{p}_j\}_{C:j}\right\}_j$$

Each agent in the community constructs the complete community signature of the proposal by which institutional-membership of agent a_i denoted with

$\{\hat{p}_i\}_C$ emerges. Agent keeps the list of the institutional-memberships to be used later in communication and verification.

Stage 3. Generation of Institutional-money. Agents perform the generation of institutional-money for each agent according to the agreement. Each institutional-money is based on assertion $\tilde{p}_i = \langle id_{\tilde{p}_i}, money, C, \{serial = 0, value = val, owner = a_i\}\rangle$ which is broadcasted by the future owner of the money, agent $a_i \in C$, to other agents:

$$(\forall a_i \in C)\ a_i \overset{\nearrow}{\underset{\searrow}{\Leftrightarrow}} C\colon \{\{\tilde{p}_i\}_{C:i}\}_i$$

Upon receiving the message, each agent $a_j \in C, a_j \neq a_i$ verifies the signature and the information contained in the proposal and checks it against the agreement. Agent a_j gives his partial signature on the proposal and sends it back to agent a_i:

$$a_j \mapsto a_i : \left\{\{\tilde{p}_i\}_{C:j}\right\}_j$$

Agent a_i who receives all partial signature of $\tilde{p}_i$ constructs complete community signature of the proposal implying that the $id_{\tilde{p}_i}$ is collectively accepted to be institutional-money denoted with $\{\tilde{p}_i\}_C$. Note that *serial* is equal to 0 (zero) since the first owner of $id_{\tilde{p}_i}$ is a_i.

At the end of this stage, every agent a_i is recognized to be the member of community C given institutional-membership $\{\hat{p}_i\}_C$. During the protocol run in stage 2, every agent learns institutional-membership of others. They keeps the list of institutional-memberships (or at least list of hash of institutional-memberships) for verification purposes. In contrast to this, every agent must keep his own institutional-money since no agent learn it from the protocol.

As explained before, agents who perform bootstrapping protocol are the pioneers Ψ. The first set of shared community key that were generated at the first stage of the bootstrapping protocol are kept by the pioneers. This set, denoted with $\{\kappa^{-1}_{C:1}...\kappa^{-1}_{C:n}\}$, will be used by the pioneers to re-share the community private-key in the event of the expelling of a member.

6.2.5 Enrolling New Member

As community grows, there is occasion when new candidate wants to join the community. Enrollment can be done either with recommendation from existing member or the new member sells goods or services to existing member to receive institutional-money. The audition process of new candidate is not the issues discussed here, since this chapter focuses on the technical details. Particularly, how shared communal private-key can be re-shared in such way that the new member receives his shared

Algorithm 6.1 Enrolling new member. Part 1

- **Before :**
  - Community C consists of n agents $\{a_1 ... a_n\}$
  - The structure of community private-key $\left\{k_{C:1}^{-1} ... k_{C:n}^{-1}\right\}$ constructed from $\{\langle d_{C:1}, N_C\rangle ... \langle d_{C:n}, N_C\rangle\}$ satisfying $d_C = \sum_{i=1}^{n} d_{C:i}$. Each agent $a_i \in C$ holds $k_{C:i}^{-1}$
  - The new candidate $a_{n'}$ assumed to know community public-key k_C
- **After :**
  - New community structure C' consists of n'agents $\{a_1 ... a_n, a_{n'}\}$, where $n' = n + 1$
  - New structure of community private-key $\left\{k'^{-1}_{C:1} ... k'^{-1}_{C:n'}\right\}$ constructed from $\{\langle d'_{C:1}, N_C\rangle ... \langle d'_{C:n}, N_C\rangle\}$ satisfying $d_C = \sum_{i=1}^{n} d'_{C:i}$. Each agent $a_i \in C'$ holds $k'^{-1}_{C:i}$
  - Institutional-memberships for $a_{n'}$: $\{\hat{p}_{n'}\}_C$

Assume that communication channels are private and agents perform the protocol correctly.

1. Each agent $a_i \in C$ chooses set of n' random positive integers $\left\{\delta_1^i ... \delta_{n'}^i\right\}$ satisfying $\delta_1^i + ... + \delta_{n'}^i = d_{C:i}$ and sends each of them to other agents including the new candidate:

$$\left(\forall a_i, a_j \in C' \mid a_i \neq a_j \mid a_i \neq a_{n'}\right) : a_i \mapsto a_j : \left\{\delta_j^i\right\}_i$$

   and keeps the δ_i^i for himself for next step.

2. Upon receiving all integers sent in step 1, each agent $a_j \in C'$ including the new candidate constructs new $d'_{C:j}$ such that

$$d'_{C:j} = \delta_j^1 + ... + \delta_j^n$$

   At the end of this phase, new structure of partial community keys is $\left\{k'^{-1}_{C':1} ... k'^{-1}_{C':n'}\right\}$ constructed from $\left\{\langle d'_{C:1}, N_C\rangle ... \left\langle d'_{C:n'}, N_C\right\rangle\right\}$

Algorithm 6.2 Enrolling new member. Part 2

3. The candidate $a_{n'}$ constructs an assertion $\hat{p}_{n'}$ which contains his public-key, partially signs it with his shared communal private-key ${k'}^{-1}_{C':n'}$ and proposes it to the community
$$a_{n'} \overset{\nearrow}{\underset{\searrow}{\leftrightarrow}} C\text{: } \{\{\hat{p}_{n'}\}_{C':n'}\}_{n'}$$
4. Upon receiving the proposal, each agent $a_i \in C', a_i \neq a_{n'}$ partially signs $\hat{p}_{n'}$ using his new ${k'}_{C:i}$ and broadcast it to the community
$$(\forall a_i \in C' | a_i \neq a_{n'}) : a_i \overset{\nearrow}{\underset{\searrow}{\leftrightarrow}} C\text{': } \{\{\hat{p}_{n'}\}_{C:i}\}_i$$
by which all agent can learn the institutional-membership of $a_{n'}$, namely $\{\hat{p}_{n'}\}_C$, to be appended in their database.

From this state, all agents work with new set shared community keys $\left\{{k'}^{-1}_{C:1} \ldots {k'}^{-1}_{C:n'}\right\}$.

communal private-key and without revealing the community private-key.

Algorithm 6.1 followed by second part in 6.2 illustrated in figure 6.3, demonstrate how agents collaborate to re-share the structure of community private-key such that all members, including the new one, receive new part of shared community private-key to work with. Despite of its simplicity, the algorithm keeps community private-key k_C^{-1} secret and allows no agent learn new shared community key of others. After having new structure of shared community private-key, the new agent proposes assertion for his institutional-membership to be used later in the transaction. This protocol maintains the value of private-key component of community private-key d_C which is proved as follows:

Proposition 6.1: *Protocol 6.1 (step 1 & 2) causes state transition that maintains the value of community-key d_C*

Proof: By definition, $d_C = \sum_{i=1}^{n} d_{C:i}$. In step 1, $d_{C:i}$ is splitted such that $d_{C:i} = \sum_{j=1}^{n'} \delta_j^i$, and thus $d_C = \sum_{i=1}^{n} \sum_{j=1}^{n'} \delta_j^i$. Let d'_C be the value of d_C after transition where $d'_C = \sum_{j=1}^{n'} d'_{C:j}$. By definition of step 2, $d'_{C:j} = \sum_{i=1}^{n} \delta_j^i$ and thus $d'_C = \sum_{j=1}^{n'} \sum_{i=1}^{n} \delta_j^i$. It can be concluded that $d'_C = d_C$. ∎

6.2.6 The Expelling of a Member

When an agent leaves community for some reasons, shared community private-key losses one of its part which had been hold by agent who leaves. Therefore, there should be a way to restructure it. This is the occasion where the pioneers who assumed never leave the community play their important part.

Algorithm 6.3 The Expelling of a Member

- **Before**
  - n agents in community $C : \{a_1 ... a_n\}$ in which there are set of pioneers agents $\Psi \subseteq C$ where ν is the sum of pioneers $\nu = |\Psi|, \nu < n$
  - set of current shared community private-key $\left\{k^{-1}_{C:1} ... k^{-1}_{C:n}\right\}$
  - set of the first shared community private-key $\left\{\kappa^{1}_{C:1} ... \kappa^{1}_{C:\nu}\right\}$
  - agent who will be convicted $a_v \in C$ where $a_v \notin \Psi$
- **After**
  - new community structure : $C' = C \setminus \{a_v\}$ and $n' = n - 1$
  - set of new shared community private-key structure $\left\{{k'}^{-1}_{C:1} ... {k'}^{-1}_{C:n'}\right\}$

Let $\left\{d^1_{C:1} ... d^1_{C:\nu}\right\}$ be private component of $\left\{\kappa^{-1}_{C:1} ... \kappa^{-1}_{C:\nu}\right\}$ such that $d^1_C = d^1_{C:1} + ... + d^1_{C:\nu}$ where d^1_C is community RSA private-key component of κ^{-1}_C.

1. Each agent $a_i \in \Psi$ chooses n' random positive integers $\left\{\delta^i_1 ... \delta^i_{n'}\right\}$ such that $\delta^i_1 + ... + \delta^i_{n'} = d^1_{C:i}$, and sends each of them to other agents
$$\left(\forall a_i, a_j \in \Psi \times C', a_j \neq a_i\right) : a_i \mapsto a_j : \delta^i_j$$
and keeps δ^i_i for himself.
2. Upon receiving the integers sent in step 1, each agent $a_j \in C'$ constructs his new shared communal private-key component ${d'}_{C:j}$ such that $d'_{C:j} = \delta^1_j + ... + \delta^j_j + ... + \delta^\nu_j$. Hence, each agent $a_j \in C'$ has new shared communal private-key ${k'}^{-1}_{C:j} = \left\langle d'_{C:j}, N_C \right\rangle$. From this state, agents use the new structure of shared community key $\left\{{k'}^{-1}_{C:1} ... {k'}^{-1}_{C:n'}\right\}$.
3. Each agent $a_j \in C'$ purges a_v's institutional-memberships $\{\hat{p}_v\}_C$ from his knowledge.

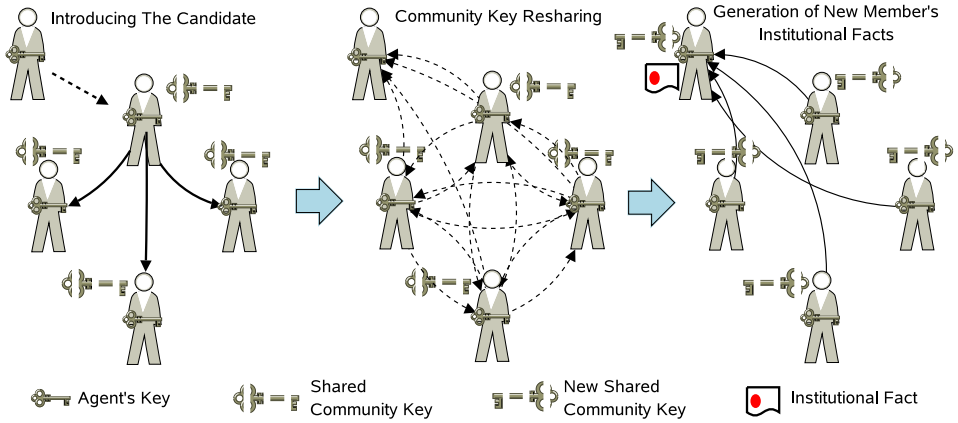


Figure 6.3: Illustration of Algorithm in Enrolling a New Member

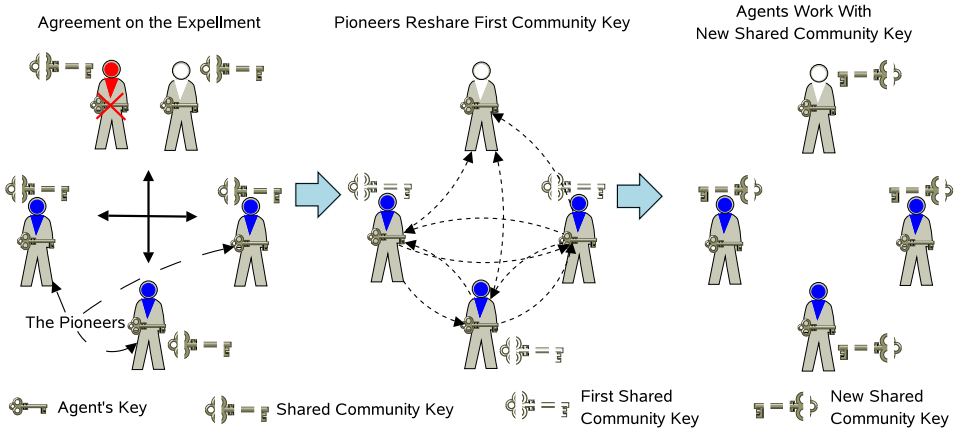


Figure 6.4: Illustration of Algorithm in Expelling a Member

Algorithm 6.3 that is illustrated in figure 6.4 demonstrates restructuring process of shared communal private-key. It takes advantage of the pioneers who hold the first structure of shared community private-key $\{\kappa_{C:1}^{-1}...\kappa_{C:n}^{-1}\}$. Basically, the algorithm is simple, each pioneer divides his part of κ_C^{-1} and distributes them to the rest of n' agents in the community, where $n' = n - 1$. At the end of the process the new structure of shared community private-key is formed with which agents work from this point.

Given the proof of algorithm in enrolling new member (proof 1) , the value of private component of shared community private-key in any state of the system should be equal to the value of d_C^1 from the first structure of shared community-key $\{\kappa_{C:1}^{-1}...\kappa_{C:n}^{-1}\}$. Algorithm of expelling the member is derivativ of algorithm of enrolling new member resharing the first d_C to all members that are left in community. Hence, the value of d'_C (after transition) is equal to d_C^1 as well as d_C

(before transition).

The two resharing protocols (Enrolling New Member and The Expelling of a Member) keep the value of d_C as constant in any state of the system eventhough structures of shared community private-key are changed during these transitions. It delivers advantage to the system such that communty signature verification can be done without altering community public-key k_C.

6.3 Transaction Protocol

Recalling the design issues and intended solution addressed before, here are specific objectives to be achieved by the transaction protocol:

1. Achieving authentication and Multiparty Non-Repudiation
2. Preventing double spending
3. Achieving m-out-of-n majority
4. Reducing communication cost of original transaction algorithm

As explained in earlier section, the use of digital signatures, distributed cryptography, and virtual hierarchy establishes authentication and Multiparty Non-Repudiation. Double spending can be prevented by incrementing serial field in the institutional-money during every transaction in order to identify the recentness of the institutional-money. Hence, the real owner of the institutional-money can make a rejection statement during the transaction along with the proofs of the possession of particular money.

Preventing double spending can also be achieved by collaborately maintaining the list of eliminated or revoked institutional-money. Earlier design of the protocol [Tarigan, 2006] employs revocation list. It was found that it is costly to maintain the community signed revocation list. In the event of transaction, all agents should broadcast his partial signature to others signifying his acceptance of the revocation list. With this, all agents can learn community signature on the revocation list. Hence, the cost would be equal to the cost of algorithm presented in chapter 3.

Furthermore, it is difficult to handle multiple transactions at one time, since agents can only work on one revocation list at a time. The thinkable solution to handle multiple transactions with revocation list is to have window of transactions in which multiple transaction proposals are "buffered" and finished after certain time. However, further research lead to the design proposed in this chapter in which no revocation list is necessary.

Threshold cryptography, a variant of distributed cryptography, offers a solution how to achieve m-of-n majority in the community signature such that m agents,

where $m < n$, can collaborate to construct community signature on a message. This algorithm is designed mainly to achieve robustness of the system solving non-reliable communication infrastructure or non-reliable agents. However, the protocol does not employ this algorithm to achieve m-of-n majority signature. Since, double spending prevention requires all agents to be present and to check proposals against their currently owned institutional-money. To solve m-of-n majority, trading agents perform so-called majority objection to the minorities. This will be explained later in detail.

The following descriptions offers the known best-practice solution to these issues. It starts by explaining the anatomy of transaction proposal.

6.3.1 Anatomy of Transaction Proposal

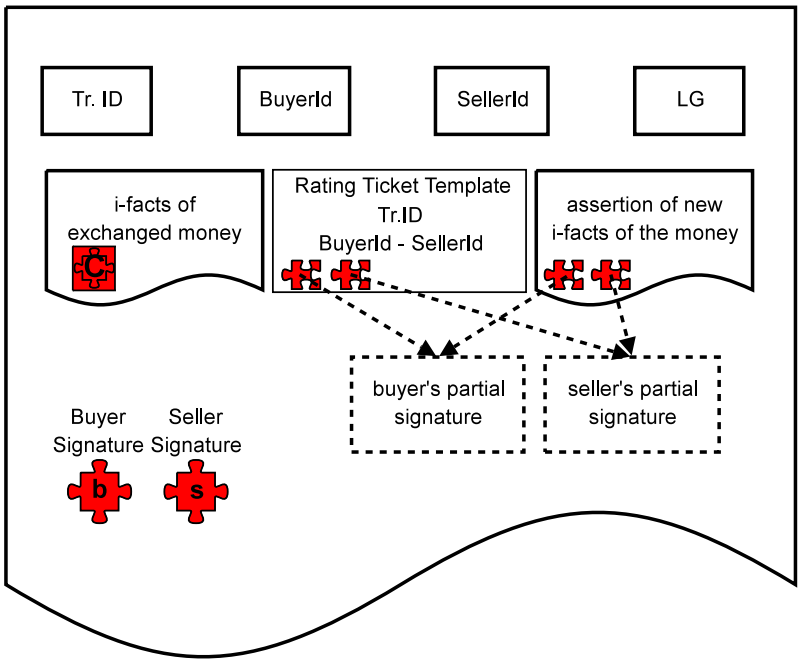


Figure 6.5: Anatomy of a Transaction Proposal

Refer to the transaction algorithm presented in chapter 3, traders initiate a transaction by sending transaction proposal $\pi = \langle P_n, P_o \rangle$ to the community. The structure of transaction proposal is extended in this chapter to meet requirement of transaction protocol. Figure 6.5 presents pictorial view of a transaction proposal in which several components are attached and signed by both traders, the buyer and the seller. Formal description of transaction proposal π is:

$$\pi = \left\langle id_\pi, a_b, a_s, LG, \{\tilde{p}_z\}_C, \{\tilde{p}'_z\}_{C:b,s}, \{rt_\pi\}_{C:b,s} \right\rangle$$

where id_π is unique id of the proposal which is also the id of the transaction, a_b and a_s are buyer-id and seller-id respectively, LG is the loss-gain ratio arbitrarily given by buyer, $\{\tilde{p}_z\}_C$ is institutional-money belongs to a_b to be exchanged with goods or services from a_s, $\tilde{p}'_z$ is proposed assertion for overriding institutional-money $\{\tilde{p}_z\}_C$, and $rt_\pi = \{id_\pi, a_b, a_C\}$ is rating-ticket for this transaction to be used later in rating propagation. Note that $\{\tilde{p}_z\}_C$ and $\{\tilde{p}'_z\}_{C:b,s}$ represent the elements of transaction proposal P_n and P_o.

If $\tilde{p}_z = \langle id_{\tilde{p}_z}, money, C, \{serial = s, value = v, owner = a_b\}\rangle$ is the current assertion of exchanged institutional-money then the proposed assertion should be $\tilde{p}'_z = \langle id_{\tilde{p}_z}, money, C, \{serial = s + 1, value = v, owner = a_s\}\rangle$. In the new assertion, identification of current owner a_b is replaced with new one a_s, the values of *value* and $id_{\tilde{p}_z}$ should be equal to the originals, and the *serial* value is incremented by one. Both traders should give their partial community signatures on the assertion as well as on the rating ticket for which they shall be denoted with $\{\tilde{p}'_z\}_{C:b,s}$ and $\{rt_\pi\}_{C:b,s}$ respectively.

To simplify protocol explanation, it is configured that, after buyer and seller agree on the transaction, the buyer is the responsible to construct the proposal and gives it to the seller to be verified, completed, and signed. Seller gives back the proposal to be signed by buyer and the buyer broadcast the transaction proposal to the community. Here, the complete transaction proposal is denoted with $\{\pi\}_{b,s}$ for which it is signed by both traders signifying that both traders agree on π.

6.3.2 The Protocol

Protocol description is presented in form of flowchart and algorithm. Flowchart gives pictorial view to describe the basic idea and protocol description in algorithm delivers detailed actions. Figure 6.7 draws a flowchart to demonstrate of how traders initiate the transaction by broadcasting proposal to the community, how traders perform some consequence actions upon receiving replies, and finalize the transaction. Figure 6.8, 6.9, and 6.10 illustrate three flowcharts of actions performed by collaborators upon receiving different requests from the traders. Detailed steps of the protocol are presented partly, for formating purposes, in algorithm 6.4, 6.5, 6.6, and 6.7.

Transaction protocol starts with the assumption that the traders have already established agreement to have a transaction. Step 1 and 2 presented in algorithm 6.4 shows how traders construct transaction proposal π and broadcast it to the community. Step 3 presented in algorithm 6.5, describes how collaborating agent should accept or reject the proposal. The three main conditions should be fulfilled by a proposal are (i) proposal and signatures validity, (ii) not a double spending attempt, and (iii) seller can be trusted given loss gain ratio provided by buyer and what agent beliefs about the trustworthiness of seller. Upon receiving acceptance or rejection, traders perform what specified in the step 4 presented in algorithm 6.6.

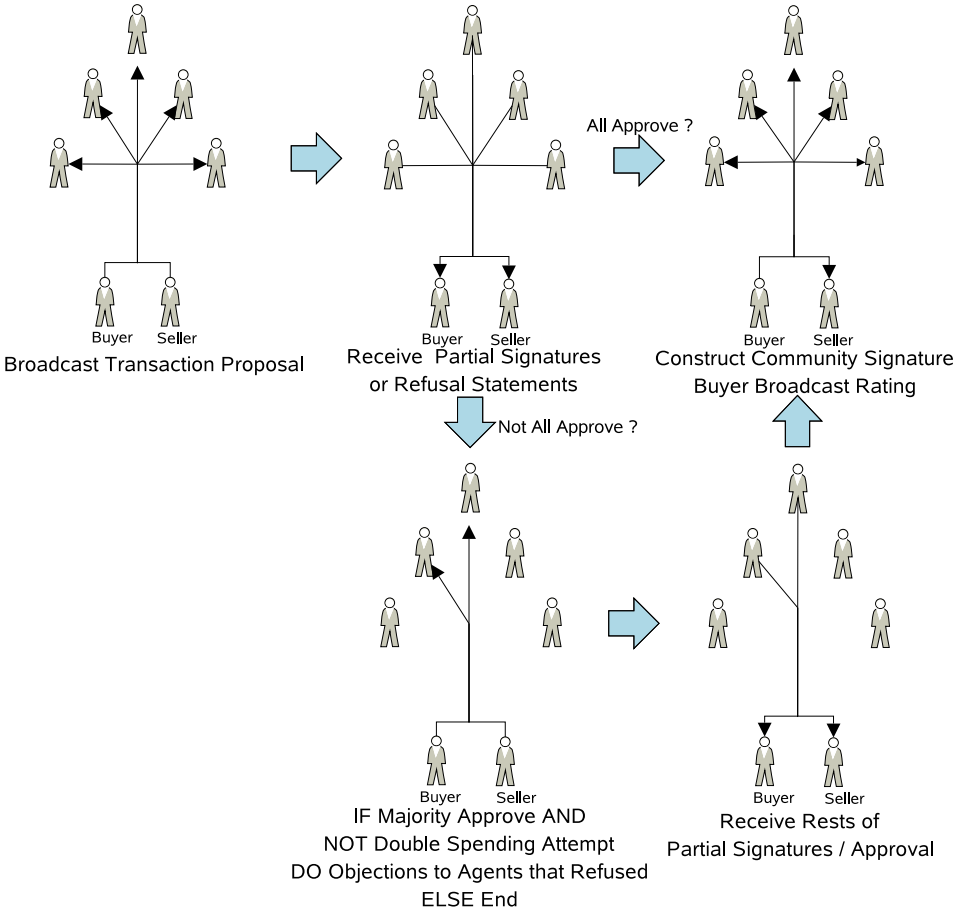


Figure 6.6: Illustration of Transaction Protocol

If all collaborators accept the proposal then traders attain all partial community signatures and thus transaction is committed. Traders finalize it by constructing new institutional-money and rating ticket. On the contrary, traders perform step 4b presented in algorithm 6.6 that basically states that traders check whether the majority m of n agents accept the proposal and none of the reason state that the proposal is a double spending attempt. If this condition applies, then traders may do the objection to the the minorities who have rejected the proposal with the proofs that majorities accept the proposal by presenting signatures signifying their acceptance. Based on the concept that majority wins, the minorities ought to accept the proposal and reply the objection with their partial signatures of the proposal and thus transaction commits. Algorithm 6.7 illustrates detailed action performed by minorities upon receiving the objection as well as action performed by traders in finalizing their transaction.

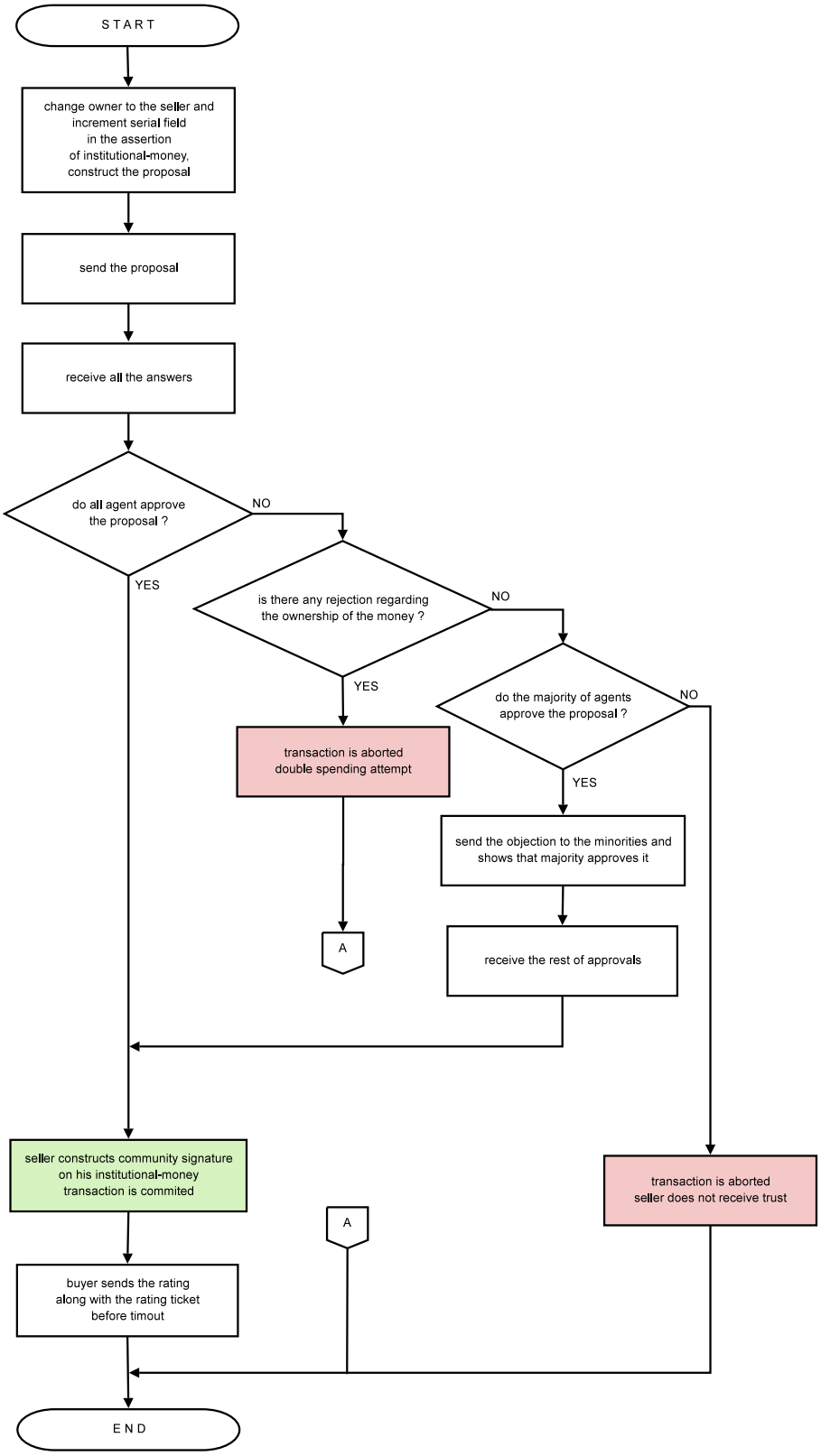


Figure 6.7: How traders initiate and end the transaction

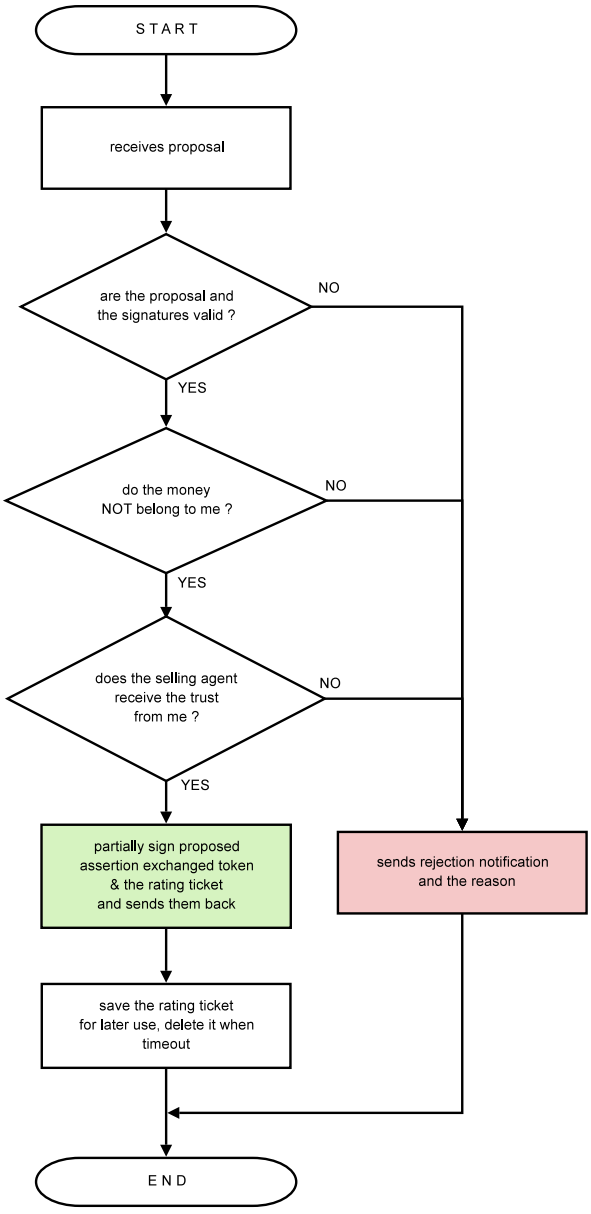


Figure 6.8: How Agents Collaborate in Receiving Proposal

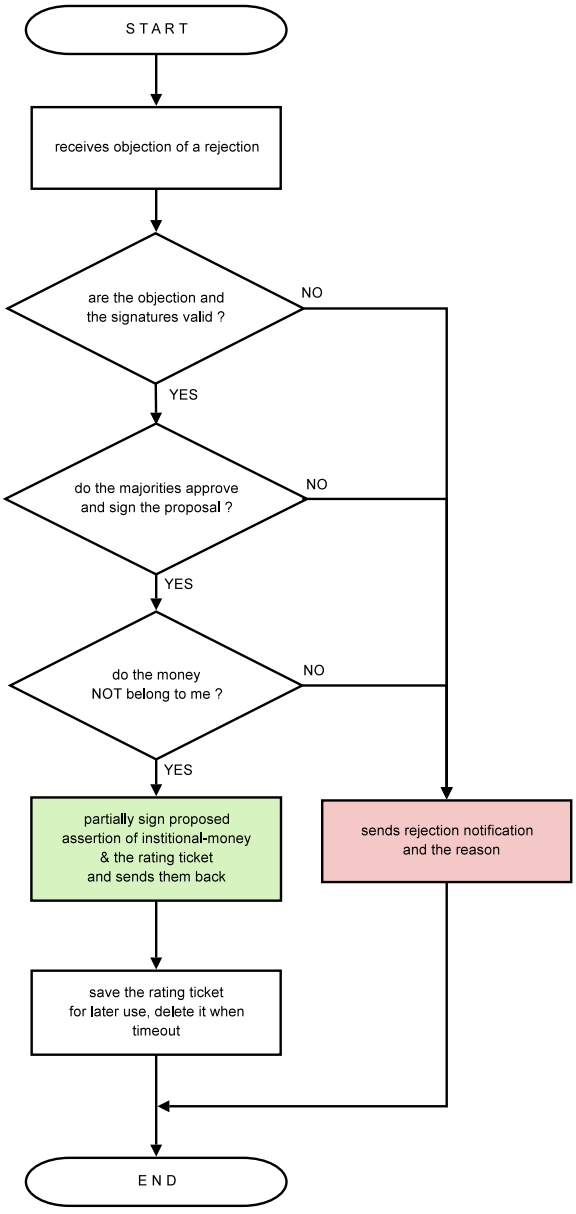


Figure 6.9: How Agents Collaborate In Receiving Objection

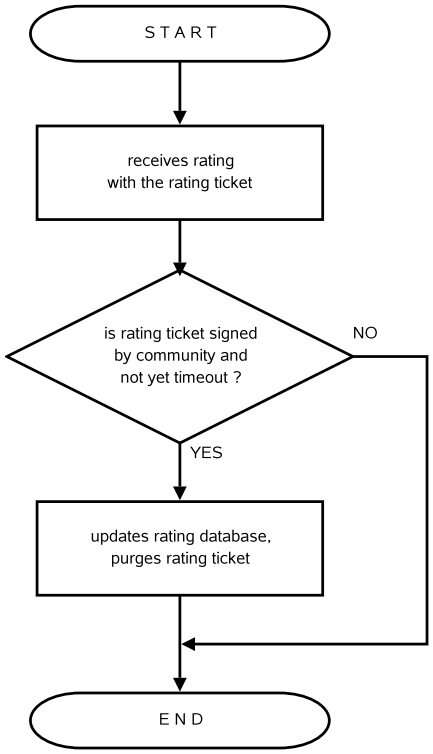


Figure 6.10: How Agents Collaborate in Receiving Rating

6.3.3 Multiple Traders Transaction

There are occasions that more than two traders should involve in a transaction. For example, a file distributor, who has high capacity bandwidth, sells a file, made by a producer, to the buyer. In this transaction, buyer pays the price of the media to the file distributor and the distributor pays license which is part of the sold price to the producer.

In order to cover this scheme, the protocol can be enhanced by a slight modification to the proposal. Let a_b, a_s, a_p be the buyer, seller, and the producer respectively who want to make a transaction in which a_b pays the media with institutional-money$\{\tilde{p}_x\}_C$ to the seller and seller pays the royalty with institutional-money $\{\tilde{p}_y\}_C$. The proposal constructed would be:

$$\pi = \left\langle id_\pi, (a_b, a_s, a_p), LG, \left(\{\tilde{p}_x\}_C, \{\tilde{p}_y\}_C\right), \left(\left\{\tilde{p}'_x\right\}_{C:b,s,p}, \left\{\tilde{p}'_y\right\}_{C:b,s,p}\right), \{rt_\pi\}_{C:b,s,p} \right\rangle$$

Without being explicitely told the roles of each traders, the collaborators learn the role of each agent in the proposal by comparing proposed new assertion with the institutional-money. And thus making the transaction simple because there is no significant difference between conventional two traders transaction with multiple

Algorithm 6.4 Transaction Protocol Part 1

Before

- institutional-facts $\{\tilde{p}_z\}_C$, where $\tilde{p}_z = \langle id_{\tilde{p}_z}, money, C, \{serial = s, value = v, owner = a_b\}\rangle$ stating institutional-money $id_{\tilde{p}_z}$ belongs to agent a_b
- agent a_s sells his goods to a_b in exchange for institutional-money $id_{\tilde{p}_z}$, and agent a_b determines the loss to gain ratio expressed in LG

After

- Transaction is Commited :
  $\{\tilde{p}_z\}_C$ is overrided with $\{\tilde{p}'_z\}_C$ which states that $id_{\tilde{p}_z}$ belongs to agent a_s
  or
- Transaction is Aborted

1. The transacting agents, a_b and a_s, constructs transaction proposal π proposal

$$\pi = \left\langle id_\phi, a_b, a_s, LG, \{\tilde{p}_z\}_C, \{\tilde{p}'_z\}_{C:b,s}, \{rt_\pi\}_{C:b,s} \right\rangle$$

   Here, $\tilde{p}'_z = \left\langle id_{p_z^0}, money, C, \{serial = s+1, value = v, owner = a_s\}\right\rangle$ is assertion of new institutional-money and rt_π is the rating ticket.

2. Transacting agents sign and broadcast the proposal to other agents

$$\{a_b, a_s\} \Diamond C : \{\pi\}_{b,s}$$

traders transaction.

6.4 Summary

This chapter has presented an implementation scenario of the framework in P2P file trading community and analysis of the issues which derives the requirements of protocol. Based on the requirement, this chapter proposes a design of P2P protocol based on the scheme presented in previous chapters which utilizes distributed cryptography algorithm. Based on the descriptions above, the protocol achieves the following items:

1. Authentication
   All messages transmitted should signed and verified using community public-key as the anchor.

Algorithm 6.5 Transaction Protocol Part 2

3. Upon receiving the proposal π, each agent $a_i \in C, a_i \neq a_b, a_i \neq a_s$ examines it to judge :

   (a) proposal validity: whether construction is valid and the signatures are correct and verified against their institutional-membership and community public-key k_C

   (b) whether this proposal is double spending attempt:

      i. whether a_i indicates that $\{\tilde{p}_z\}_C$ belongs to him (having the same id but the *serial* field stated in a_i's money is greater than it's in $\{\tilde{p}_z\}_C$)

      ii. whether a_i indicates that a_b is also having transaction with him using the same institutional-money $\{\tilde{p}_z\}_C$

   (c) a_s trustworthiness; given LG from the proposal and a_s reputation believed by agent a_i

   If both examination is succeeded then agent constructs his part of community partial signature on $\tilde{p}'_z$ as well as rt_π, and sends them back to the seller and buyer:

$$a_i \mapsto \{a_b, a_s\} : \left\{id_\phi, \left\{\tilde{p}'_z\right\}_{C:i}, \{rt_\pi\}_{C:i}\right\}_i$$

   agent a_i remembers the rt_π on his temporary memory until a_b informes the rating of π or rt_π is timeout.

   On the contrary, agent a_i replies with rejection reason $reason \in \{not\_valid, not\_trusted, double\_spending\}$:

$$a_i \mapsto \{a_b, a_s\} : \{id_\phi, reason, proof\}_i$$

   When the reason of rejection is *double_spending*, a_i attaches his institutional-money (which has serial number greater than a_b's) in field of *proof* .

---

2. Multi-party Non-Repudiation
   The utilization of distributed cryptography allows agent to perform partial community signing on assertion by which community signature on assertion signifies that the assertion is accepted as institutional-fact.

3. Authorization
   Based on the arbitrary loss gain ratio of particular transaction provided by buyer and the trustworthiness of the seller believed by collaborators, authorization is performed collectively using the concept of majority wins.

4. m of n majority

Algorithm 6.6 Transaction Protocol Part 3

4. Having all replies, transacting agents perform approval checking as follows :

   (a) If all agents approve the transaction proposal, then the complete community signature can be computed from all collected partial signatures and thus the new institutional-fact $\{\tilde{p}'_z\}_C$ and community-signed rating ticket $\{rt_\pi\}_C$ can be constructed. In this state, transaction with id id_π is committed or completed. Before rating ticket timeout, agent a_b broadcasts the rating of completed transaction along with the rating ticket to other:

   $$a_b \Leftrightarrow C : \{id_\pi, rating, \{rt_\pi\}_C\}_i$$

   every agent $a_i \in C$ who still remembers rt_π on his temporary memory checks the received rating and updates his knowledge towards reputation of agent a_s. Protocol stops.

   (b) On the contrary, if there are rejections then transacting agents check:

      i. if none of the reason of rejection is *double_spending* and the sum of approving agents are more or equal than majority, then transacting agents send objection to the minorities along with the proof. Let $\bar{c} \subset C$ be all agents who approve π and $\acute{c} = C \setminus \bar{c}$ be all agents who reject π, transacting agents construct o_ϕ as an objection for π such that $o_\pi = \langle \pi, \gamma \rangle$ where $\gamma = \bigcup_{a_j \in \bar{c}} \{psig_{C:j} \tilde{p}'_z\}$. Transacting agent broadcast this objection to the minorities $\acute{c}$:

      $$\{a_b, a_s\} \mapsto \acute{c} : \{o_{\phi\pi}\}_{b,s}$$

      ii. else, transaction is aborted. Protocol stops.

If certain conditions applies, the use of simple objection scheme and the concept that the majority wins enables traders to collect all n partial community signatures and complete the transaction.

5. Double spending prevention
   Since partial signing scheme requires all n agents to contribute in a community signature, collaborator, who owns the recent institutional-money proposed in the proposal or collaborator who is having uncommitted transaction using the same institutional-money, can vote down the proposal. Thus, double spending can be prevented.

6. Accounting
   The protocol allows accounting mechanism of institutional-money to be achieved similar to the physical based accounting of money in the real world.

Algorithm 6.7 Transaction Protocol Part 4

5. Upon receiving objection, each agent $a_i \in \acute{c}$ checks whether all partial signatures from all agents are valid and the sum of majorities is sufficient according to the agreement. Agent also checks whether the objection is the second attempt for double spending. If all conditions satisfied then agent a_i constructs community partial signature on $\tilde{p}'_z$ as well as rt_π, and sends them back to the seller and buyer as it is obligatory for him to follow what majorities had decided:

$$a_i \mapsto \{a_b, a_s\} : \left\{id_\phi, \left\{\tilde{p}'_z\right\}_{C:i}, \left\{rt_\pi\right\}_{C:i}\right\}_i$$

On the contrary, agent a_i replies with rejection reason $reason \in \{not\_valid, double\_spending\}$:

$$a_i \mapsto \{a_b, a_s\} : \{id_\pi, reason, proof\}_i$$

6. Receiving replies:

   (a) if all replies from the minorities are the approval of the transaction, then the complete community signature can be computed and thus the new institutional-fact $\{\tilde{p}'_z\}_C$ and rating ticket $\{rt_\pi\}_C$ can be constructed. In this state, transaction with id id_π is approved and committed. Before rating ticket timeout, agent a_b broadcasts his opinion of the completed transaction along with the rating ticket:

   $$a_b \overleftarrow{\Leftrightarrow} C : \{\{rt_\pi\}_C, rating, \}_i$$

   every agent $a_i \in C$ who still remembers the rt_π on his temporary memory verifies the signatures, checks the validity of the rating, and updates his knowledge towards reputation of agent a_s.

   (b) else, transaction is aborted.

Each agent keeps institutional-money on his own. In the event of transaction, the money can be verified using community signature as well as validated during double spending prevention embedded in the protocol.

7. Scalability
   Instead of using and maintaining list of institutional-facts, proposed protocol allows an agent to keep only his institutional-money and list of agents in the community. Moreover, collaborators do not need to maintain session of transaction, for example to wait for the protocol to be committed or aborted. That is, collaborators serve only request and react based on the conditions illustrated in above description. The only thing collaborators need to maintain is

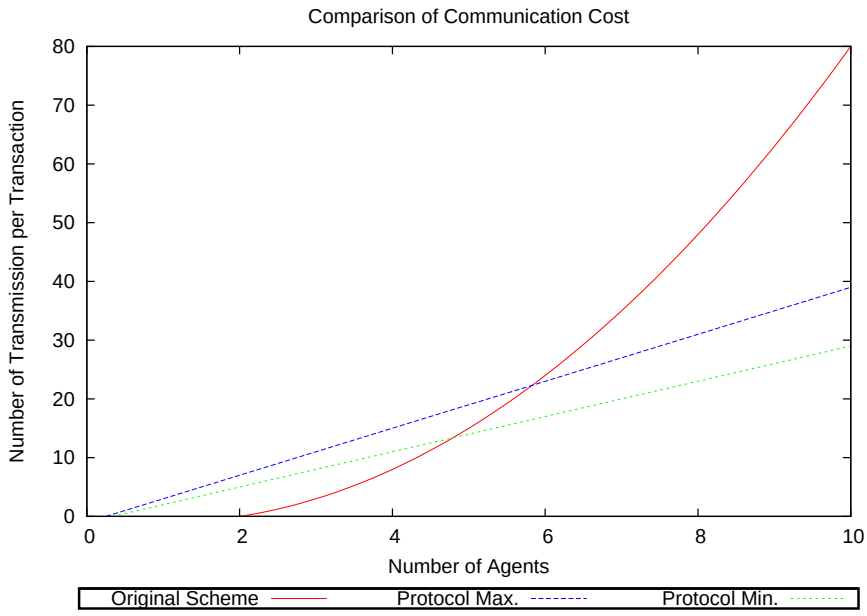


Figure 6.11: Comparison of Communication Cost between Original Transaction Algorithm and the Developed Transaction Protocol

the rating ticket which is expired after certain period of times. This eliminates the scalability problem of original transaction algorithm.

8. Reduced communication cost
   Consider broadcasting a message is equal to $n-1$ times sending message to every agent, the protocol needs minimal of $3n-1$ and maximal of $3(n+m)-1$ message transmissions per transaction. This is significant reducement from n^2-2n message transmission from original scheme. Figure 6.11 shows comparison graph of communication costs between original scheme and the protocol proposed in this chapter.

These achievements are regarded to be the best-practice solution for the framework to make feasible and plausible implementation in the future. The next chapter shows how transaction protocol developed in this chapter can be applied in P2P file-sharing network as trading infrastructure for their users.

Chapter 7

Application of the Framework

This chapter finalizes the thesis by presenting an architecture of implementation of proposed framework that provides trading infrastructure for existing P2P networks. It is the attempt to drag the framework into more realistic implementation. First, the objective and requirements of the system are presented to identify challenges and obstacles in designing desired application. Then, a section presents the architecture of the trading system as well as the integration with existing P2P systems. The next sections demonstrates the work-flow of the trading and an algorithm designed to solve problems previously discussed. This chapter ends with a section that discusses the design of the trading system and elaborates possible applications and business opportunities given the implementation of the framework.

7.1 Introduction

7.1.1 Objective

The P2P networks have been associated with free content distribution and unregulated network. Despite the fact that P2P network has distinctive advantages and wide range of utilization, these properties detains the development of P2P network toward the stage where sustainable trading or economical transactions can be conducted on the network. This chapter presents the application of the framework proposed in this thesis that enables sustainable economical transactions within existing P2P networks.

7.1.2 Issues and Requirements

In the design of the P2P transaction enabler system, there are several requirements should be fulfilled addressed because of certain characteristics of P2P Network significantly affect the implementation of the framework, namely:

Availability

The P2P network is well known to be the decentralized network in which peers are not always available to provide the services. Although today's Internet network services are becoming more affordable and reliable, most P2P users join the network on the need basis.
In the proposed framework, the completion of transaction requires all peers to be available for collaboration. Although nowadays network infrastructures, such as Jabber[1] messaging, ensure the reliability of message transport, the absence of the peers would create delay in the completion of transactions. This might become the drawback for the network. Inevitably, the architecture should allow agents to be virtually available for collaboration.

Address Management

How can an agent find other agent in the network. This problem arises since most of P2P networks use pseudo-names as identifier. Pseudo-names are cheap, reusable and can be used by any Internet users connected in the particular network. Moreover, most Internet Service Providers (ISPs) use dynamic IP address assignment or Network Address Translator (NAT). It makes difficult to determine source address of agent solely using IP address. In this respect, the architecture should support address management and address resolution scheme to overcome the problem of pseudo-name based P2P addressing.

Integration with Existing P2P Network

Every P2P network has its distinctive applications and features. For example, eDonkey and BitTorrent network provide widely used file sharing infrastructure. Internet users can take advantage from PeerCast and P2PTV network to provide multicast network for multimedia content such as movie or TV. Considering the wide-used and powerful application of the existing P2P networks, the architecture of this trading system should enable integration with existing P2P network and software.

7.2 The Architecture

7.2.1 The Overlay Trading Network

The trading network is designed as an overlay network which is superimposed on top of existing P2P networks. It provides trading infrastructure for P2P users who are member of the trading network. As shown by figure 7.1, the trading network and P2P networks are overlay networks which run upon Internet protocol. Here,

[1] http://www.jabber.org

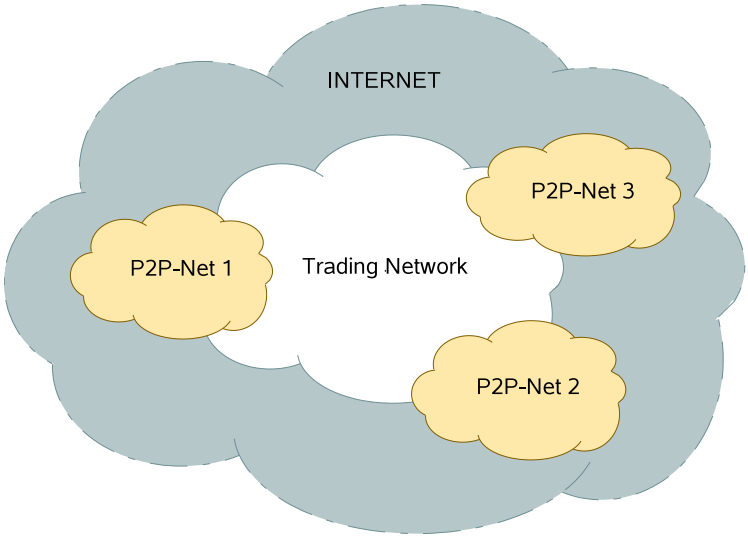


Figure 7.1: Trading Network as Overlay Network for existing P2P Networks

P2P networks provide functionality and features as they were designed for. The trading network utilizes the P2P networks as search and delivery infrastructure for the traders. With this design, traders are able to exchange any content type on which certain P2P network provides.

7.2.2 Trading Program

In order to cope with the requirement of availability and address management, the trading system incorporates two types of program in the family. The so-called User-side Program (UP) and Delegation Program (DP). The UP resides in the user's computer performing complete functionality of trading agent in community as well as collaboration related tasks to complete transaction. DP, on the other hand, is web-service program arbitrarily hosted by the agent in the reliable public web-server. DP performs subset of functionality of UP as well as address management tasks that is explained later. An agent in trading community needs both UP and DP to conduct trading.

The trading network utilizes two application protocols, namely HTTP and Jabber. DP, as explained earlier, is web-service program hosted public web-server that has fixed URL address and uses HTTP as application protocol. UP runs on the user's computer that might be connected to the Internet with dynamically assigned address through the firewall. Therefore, the Jabber protocol, for example, is suitable to be used as communication protocol between UPs. Figure 7.2 and 7.3 illustrate the anatomy of UP and DP, as well as interactions between them.

The idea in using the two programs are the following :

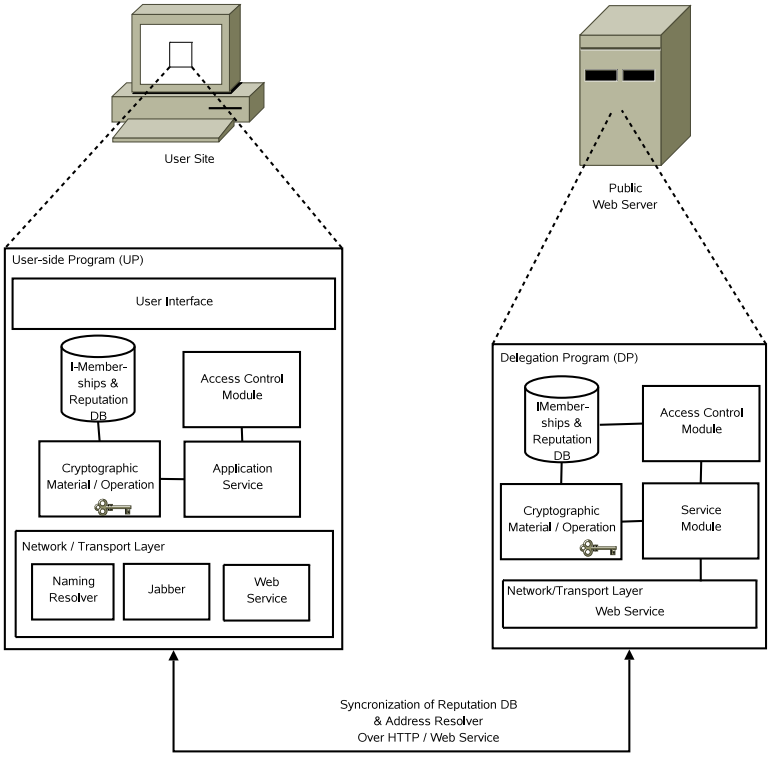


Figure 7.2: User-side Program and Delegation Program

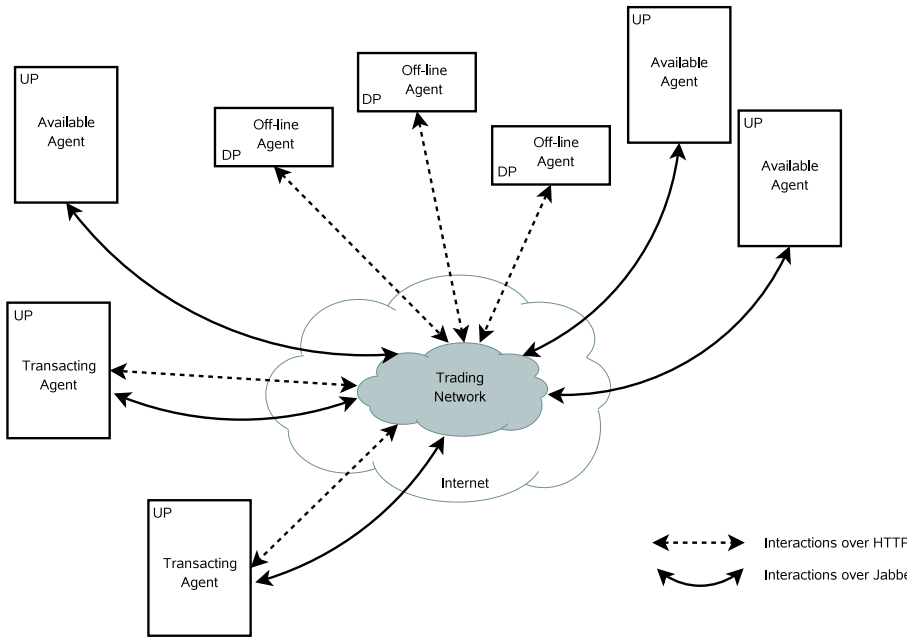


Figure 7.3: Interactions among *User-side Programs* and *Delegation Programs*

- Delegation of Collaborative Tasks
  Assume that public web servers are always available and supported by reliable network infrastructure, UP delegates its collaborative functionality to the DP when UP is not available. The tasks delegated to DP are (i) to perform partial decision trust upon request from traders, and (ii) to receive rating or feedback from others and update the rating database accordingly. That is, DP is backup peer when user goes off-line. Any update happened on DP during unavailability of UP is synchronized when UP becomes available. During database synchronization, UP updates also its current Jabber address or IP address to the DP for address resolving mechanism.

- Dynamic Address Resolving Mechanism
  Institutional-Memberships, by which agent authenticate others, contains the URL of DP which is the first place for an agent to reach the DP of queried agent. DP, which stores the Jabber address of the UP during last database synchronization, provides agent with UP's addresses. Thus, any change of UP addresses can be learned by all agents in the network.

The next section describes the functionality of both UP and DP further as well as the trading protocol utilizing both programs.

7.2.3 Integration

The P2P protocol used in inter-peers communication is specially design to carry out the service on which the network provides. Therefore, the integration of trading network with P2P network should not include the modification of existing protocol and should be transparent to the peers.

Figure 7.4 presents the integration of UP into existing P2P software. Here, UP controls the information which are fed to the Access Control List (ACL) of each peer. The ACL module in each peer orders the access to the individual resource or service which the peer provides. The seller or the service provider process the request for download or service delivery from the peer with whom seller had transaction before. After request verification the UP feeds the access control information to the P2P software to authorize download session or service utilization from the correspondent peer. The complete work-flow will be discussed in the next section.

7.3 The Work-flow

7.3.1 General Work-flow

The typical P2P network provides distributed facility for the users to advertise their resources being offered to the network. The users connects into the network

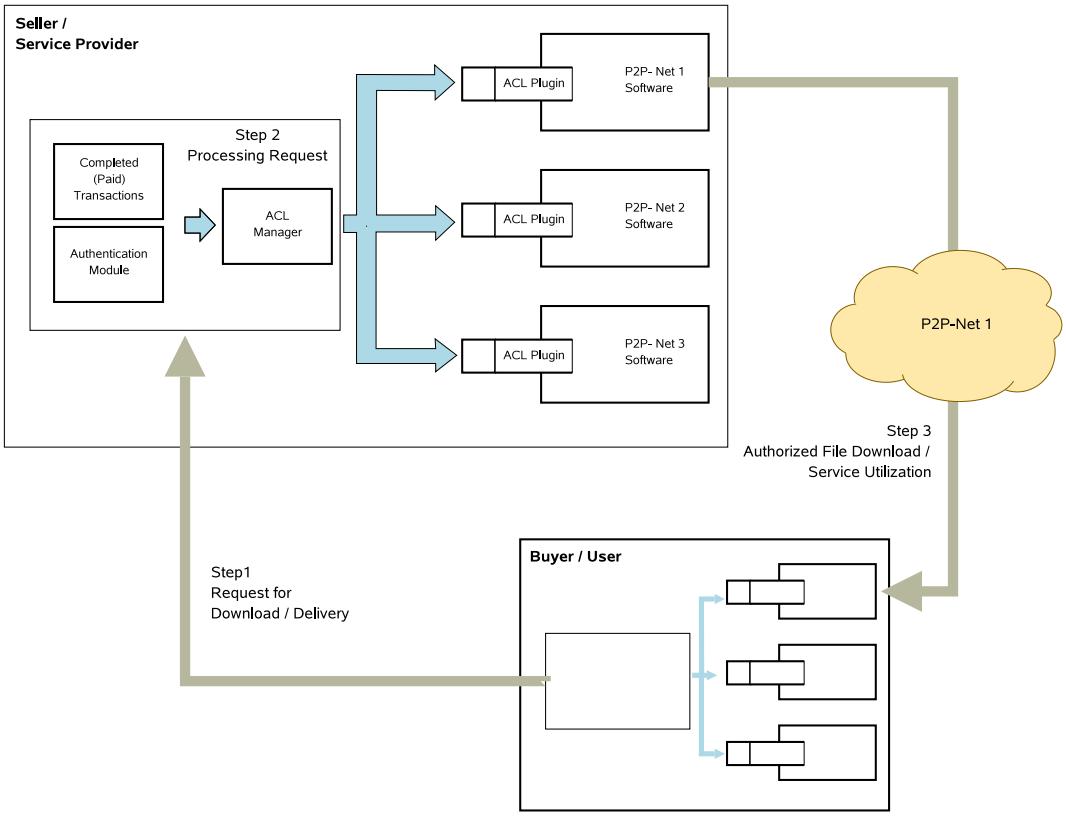


Figure 7.4: Integration of UP with existing P2P software

with ID and the IP address of the peer and the Distributed Hash Table (DHT) algorithm provides the network with peer locating and resource searching functions. Typically, user search for shared resources with the name or the peer which shares these resources.

Figure 7.5 illustrates general work-flow of the trading system. First, P2P users search the files or resources in the P2P network. The resources being offered in the trading network is tagged with special tags to identify the trading network, the ID of the resource being offered, as well as ID of the seller.

The next step is to contact the selling agent to confirm his offering and dealing the price which is done in the trading network. After price agreement, both agents completes the payment using protocol which is proposed in previous chapters. Upon receiving the transaction receipt, the buyer requests for download or service usage from the seller after authenticate him first to the seller using his ID and the transaction receipt. Note that these steps were conducted in the trading network.

Upon receiving request, the seller's UP feeds transaction information to the ACL module of P2P program which opens authorized session to the buyer to download

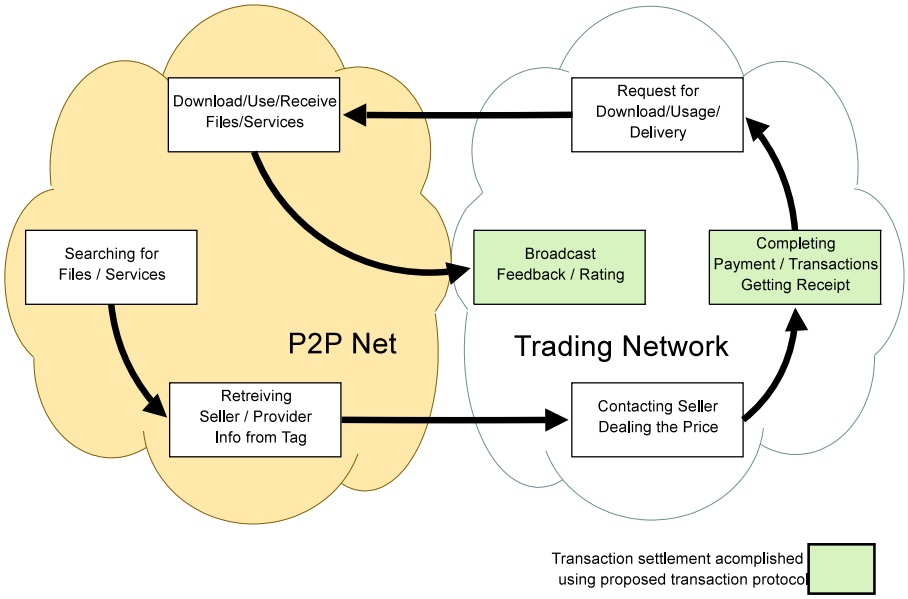


Figure 7.5: General Work-flow of Transactions

the resources or to use the services. Finally, after receiving the resources from the seller, buyer gives his feedback to the trading network resulting the change of seller reputation.

7.3.2 Address Management

One of the important issues in the design is address management. As mentioned before, the peer tends to have dynamically assigned address when the node is connected to the network. The space of ID in the P2P network is not protected and authenticated, which means that everyone can use the same ID when it is not occupied. The trading network, however, requires that an agent has specific ID because the intention to sustain the trading resources and the trust among the agents.

To solve this problem, the architecture incorporates Delegation Program (DP) which is hosted in the public web-server and therefore has fixed address in form of URL (Unified Resources Location). The URL of the DP is contained in the Institutional-Membership of the agent. Recall the description of the protocol in previous section, the Institutional-Membership is maintained by all agents in the community. Hence, DP is the fixed location where agents resolve each other's addresses.

Figure 7.6 illustrates the address or ID resolving process. Consider an agent a_b intends to contact agent a_s in the network for resource retrieval a_b has bought from a_s. First, a_b retrieves a_s's Institutional-Membership from his database to get a_s's DP URL. Assume that a_s is on-line and logs into the trading network as well

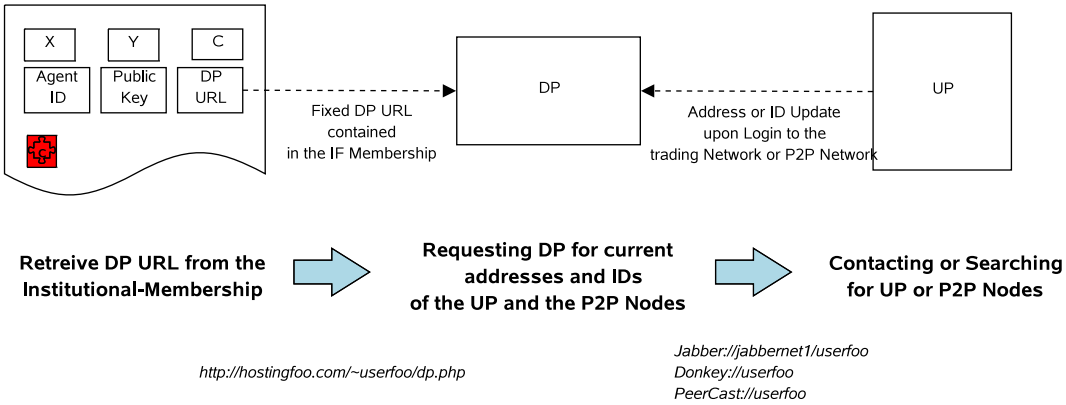


Figure 7.6: Dynamic Address Resolution

as the P2P network with his UP and P2P software respectively. a_s's UP updates his present IDs and IP addresses to his DP. Thus, a_b's UP can learn the present assigned IDs and addresses of a_s's UP by contacting a_s's DP.

7.3.3 Delivery Protocol

After transaction or payment is completed, it is time for the buyer to ask for resources or services that he paid for. Figure 7.7 illustrates the protocol by which authorized delivery can be done using a normal P2P network. The protocol encompass two phases, first is request for delivery and authentication which is done in trading network using buyer's UP and DP as well as seller's UP, and second is the delivery itself which is done in particular P2P network.

Assume that transaction or payment is completed and buyer got his ticket for what his paid. First, buyer UP requests and authenticates himself to the seller UP by presenting the ticket signed With his public-key. Seller UP verifies the signed request and requests for buyer's IDs and addresses to the buyer DP which seller trusts. Having the IDs, seller UP inserts the correspond buyer's ID into P2P node ACL which opens authorization for the buyer to grab the deliverables. At the same time, seller UP acknowledges the authentication and send the requested resources along with its location or address in P2P network. Buyer starts to download or use the exchanged resources from seller P2P node after authorized of having the right ID or address.

Validating buyer's ID from buyer's DP is important to tackle reply attack which can be launched by any peers in the network, since peers can join the network using any pseudo-name which is still available to use.

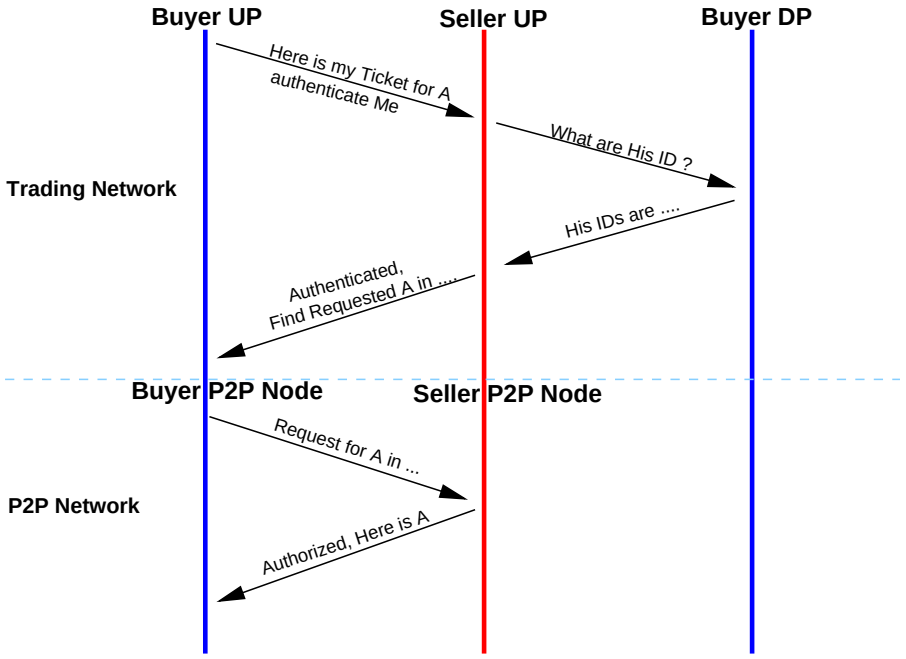


Figure 7.7: Authorized Delivery Protocol

7.4 Discussion

7.4.1 Slight Review

The application architecture presented in previous sections demonstrates how the framework can be implemented and integrated into existing P2P network. The trading network is designed as overlay network which runs on top of Jabber messaging network as well as HTTP protocol, exploiting their features and advantages.

The use of User-side Program (UP) and Delegation Program (DP) in the system enables dynamic address/ID resolution mechanism and increasing the availability of agent in the network. Using this scheme, agents can easily join any available Jabber network, - or any overlay and reliable messaging transport network, using any available pseudo-names long as they can communicate each other using the network. Here, DP is also responsible for delegating the UP when UP is not available for collaboration. Thus, the problem the availability of the network can be solved with this mechanism.

The design of the UP as ACL module for the existing P2P software enable authorized delivery of resources being exchanged in the network. The agents takes advantage dynamic address / ID resolution to get each others' addresses or IDs with which they join P2P networks. Resolved ID and addresses are used in the ACL module to authorize download or usage request from the network. Thus, it creates

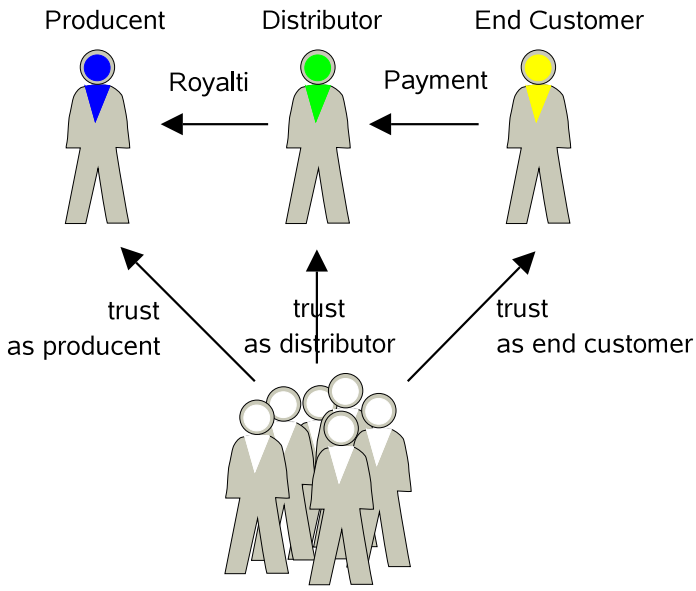


Figure 7.8: Illustration of Royalti Based Transactions

layer of authentication across P2P networks.

7.4.2 New Possibilities of Trading Scheme

Proposed architecture opens some possibilities to conduct various schemes of trading. For example:

- The framework enables the completion of multiple traders transaction which offers open accountability for the traders. Cooperation can be setup between distributors and producers to deliver the resources using royalti scheme. For example, musician is able to let the music he produces to be distributed by the set of distributors who have better bandwidth without loosing the accountability. Figure 7.8 illustrates how payment, from buyer to distributor, and royalti, from distributor to producer, can be completed in one transaction. In this illustration, trust on the three traders can be made differently according to the context of each roles in the trading.

- Currency exchange. Since transaction is based on exchanging community money, agent who has high reputation could establish money exchanger services for others. The particular agent sells "real money" to others in return of institutional-money. Moreover, agent can perform loan service by buying signed promise from other in return of institutional-money.

7.5 Summary

This chapter presented an example of application of the proposed framework, namely the trading infrastructure for existing P2P file-sharing networks. It demonstrates that the application can be integrated seamlessly into existing system by developing overlay trading network for P2P users. It also solves some problems that arise due to the nature of decentralized network. At the end some of thinkable trading schemes are discussed to give the picture of new possibilites that can be ellaborated due to the deployment of proposed architecture.

Chapter 8

Conclusion and Future Works

> *"We can-we must-choose what kind of cyberspace we want and what freedoms we will guarantee. These choices are all about architecture: about* ***what kind of code*** *will govern cyberspace, and* ***who will control it****" (Lawrence Lessig, 1999)*

8.1 General Conclusion

The uncontrollability nature of cyberspace challenges research communities to develop algorithms to secure digital-transactions and to enhance the controllability of the network. Chapter 2 is the reminder, that us, computer scientists, consciously or not-consciously had imitated social interactions into the development of those algorithms. Ad-Hoc approaches is representation of Barter exchange that solves specific, instantaneous, and non-continuous transactions happening everyday. The single authority model imitates government-to-people relationships resulting controllable digital-transactions systems found in today's E-commerce. The third one is the attempt to mimic people-to-people relationships in which control is performed not by the government, but by people, from people, and for people.

This thesis had proposed a novel framework that belongs to the third attempt: the development of algorithms and architecture that enable decentralized agents to collaborate in self-managing electronic transactions, without assuming trusted third party administering the transaction. The framework comprises the transaction algorithm that promotes collaboration to complete transactions, as well as collective authorization scheme that establishes social control for mutual security. The development of the framework is perfected by proposing a transaction protocol based on original transaction algorithm in which distributed cryptography is successfully applied to achieve mutual authentication as well as to meet best-practice requirements.

8.2 Summary of Contributions

Chapter 3 formalizes collaborative actions in the creation and elimination of institutional-facts as the basis transactions. Collective acceptance of arbitrary count-as assertions creates the social-reality of accounts of finances of trading agents. The process of collective acceptance gives deontic power to the social-object and its attributes as stated in the assertion. Transaction is conducted by collectively altering those assertions, in order to change financial statuses of transacting agents. Here, transaction logic infers developed transaction algorithm that specifies actions of agents in completing transactions.

Chapter 4 develops an algorithm so-called collective authorization that is embedded into the transaction algorithm presented in chapter 3. The objective is to establish social control allowing agents to collectively filter possible bad outcome of transactions performed by dishonest agents. Continual disappointments resulted from such outcomes might lead to perturbation of trust that disturb the sustainability of collaboration and cooperation on which this framework heavily relies. In this scheme, each agent rates the outcome of every transaction he just had and broadcasts the concluded rating from which others can learn more about trustworthiness of particular selling agent. Community authorization is performed by agents using majority based voting. Each vote is reasoned by seller reputation as well as expected loss and gain of the transaction. The algorithm enables agents as a whole to govern transactions in individual level. Hence, this is regarded to be significant contribution since known approaches of authorization in decentralized systems allow only local decision which can not guarantee that such decision comply with system-wide policy.

In order to study how it works, communal authorization scheme is studied using simulation presented in chapter 5. The objective of this simulation study is to have preliminary views of how the algorithm works as well as to identify its limitations. The simulation shows that the algorithm allows agents to learn to identify dishonest agents and prohibit them to conduct transaction. Since the idea of natural selection is incorporated into simulation algorithm, it demonstrates that trading behavior of dishonest agents are socially limited and thus decreases their population in community. Here, good behavior is promoted, bad behavior is discouraged, and thus perturbation of trust can be avoided. Furthermore, the results from parameter sweeping experiment yield a narrow working range of collective authorization. This result is regarded useful in future implementation of the framework.

Chapter 6 contributes set of transaction protocols that are developed based on original transaction algorithm. It successfully applies distributed cryptography system into the protocol to establish group authentication as well as so-called Multi-party Non-Repudiation. It consists of bootstrapping protocol performed during

establishment of community, community-key re-sharing protocols performed during the enrollment or expelling of member of community, as well as the transaction protocol itself. The achievement of the development in this chapter is that the protocol significantly reduces communication cost per-transaction compared to that from original transaction algorithm, solves the problem of scalability, and thus meets best-practice requirements.

Additionally, chapter 7 exemplifies an application of the framework. A trading infrastructure for P2P file-sharing networks is designed to enable P2P users to conduct sustainable transactions. The software architecture covers integration technique with existing P2P systems, as well as, designing web-service based delegation program to increase the availability of agents for collaboration. Furthermore, address resolving protocol and authorized-file-delivery protocol are developed to cope with practical problems of P2P. Finally, some new possibilities of applications are also discussed in this chapter.

8.3 Future Works

This thesis conveys the research into the state where decentralized trading agents are able to establish the system of communally governed transactions as well as delivers the result of preliminary study on the implementation. The ultimate milestone would be to implement the framework into a real world as a system that is usable by Internet users. Therefore, further research should be conducted to understand more about the behavior of the system, to study the acceptance of the concept by real users, and to be more certain that the architecture delivers trustworthy design. Some of the works that should be completed to achieve intended milestone are the following:

1. In depth analysis of possible attacks
   The most dangerous attack on a system would be from the insiders. The proposed protocol assumes that agents are trustworthy in the collaboration. This assumption is taken considering that agents are able to learn to identify un trustworthy agents as well as to self-organize punishment through the collective authorization protocol. However, as it is shown in chapter 5, the learning process needs certain amount of time. Therefore, further in-depth attack analysis should be performed to study all possible attacks. The results would be used to refine and enhance the protocol.

2. Formal verification of the protocol
   The work in thesis had incorporated informal proof of some part of algorithm. Nevertheless, formal verification of the whole protocol should be done in order to be sufficiently certain that the whole architecture delivers trustworthy

services as intended. In this work, some security properties should be verified using the right tools. BAN Logic Abadi et al. [1996], for example, provides framework to formally proof authentication property of the protocol. The TLA+ hierarchical proof and TLC model checker can be used to verify the correctness of the protocol. The result is taken as considerations in revising the protocol.

3. Study on Collective Authorization
   Collective authorization in this thesis takes Beta reputation and simple average as method to compute trust. Further study on using other trust computation method is necessary to understand the influence of those methods in the system. This work should include not only simulation of the system, but also the study and analysis using Evolutionary Game Theory in order to see how system would evolve in the long run. In connection with Coleman's model of decision trust, other variables such as temptation to abuse trust, that is based on LG ratio, should be also incorporated in the analysis.

4. Economic Model
   Establishment of commerce in P2P opens research spaces in developing and analyzing economy model that can be applied in this environment. The result from this area is beneficial for this research to look for modified scheme or even new scheme of transaction that is considered best-fit in the P2P environment. Furthermore, socially constructed trading network such as proposed in this thesis is mainly endorsed by community instead of world of business. Nevertheless, their involvement would be the catalisator of the use of the trading network. Thus, one of the objective in this area is to find business models and opportunities that attract business entities to involve in setting-up the trading infrastructure for mutual benefit.

Bibliography

Martin Abadi, Michael Burrows, and Roger Needham. A logic of authentication, from proceedings of the royal society, volume 426, number 1871, 1989. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press*. 1996. URL citeseer.nj.nec.com/burrows90logic.html.

Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *33rd Hawaii International Conference on System Sciences-Volume 6*, January 2000.

Ross J. Anderson. *Security Engineering: a guide to building dependable distributed system*. John Wiley & Sons Inc., 2001.

Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/1041680.1041681.

Alberto Artosi, Antonio Rotolo, and Silvia Vida. On the logical nature of count-as conditionals. In C. Cevenini, editor, *The Law and Electronic Agents: Proceedings of the LEA 04 workshop*, pages 9–34, 2004. URL http://www.lea-online.net/publications/01countas.pdf.

Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations*, page 139, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2169-X. doi: http://dx.doi.org/10.1109/CSFW.2004.7.

M. Ben-Or, O. Goldreich, S. Micali, and R.L. Rivest. A fair protocol for signing contracts. *Information Theory, IEEE Transactions on*, 36(1):40–46, Jan. 1990. doi: 10.1109/18.50372. URL http://ieeexplore.ieee.org/Xplore/url=/iel1/18/1841/00050372.pdf.

M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173, 6-8 May 1996. doi: 10.1109/SECPRI.1996.502679.

M. Blaze, J. Feigenbaum, and J. Ioannidis. Rfc 2704 - the keynote trust-management system version 2. Network Working Group, 1999a. URL http://www.faqs.org/rfcs/rfc2704.html.

Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The role of trust management in distributed systems security. pages 185–210, 1999b. URL http://www.cs.yale.edu/homes/jf/BFIK-SIP.pdf.

Manuel Blum. How to exchange (secret) keys. *ACM Trans. Comput. Syst.*, 1(2): 175–193, 1983. ISSN 0734-2071. doi: http://doi.acm.org/10.1145/357360.357368.

Dan Boneh and Matthew Franklin. Efficient generation of shared RSA keys. *Journal of the ACM (JACM)*, 48(4):702–722, 2001. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/502090.502094.

Vincent Willem Buskens. *Social Networks and Trust*. PhD thesis, Interuniversity Center for Social Science Theory and Methodology, 1999.

Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. Referee: trust management for web applications. *World Wide Web J.*, 2(3):127–139, 1997. ISSN 1085-2301.

James Samuel Coleman. *Foundations of Social Theory*. The Belknap Press of Harvard University Press, 1990.

George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems. Concept and Design*, volume Third Edition. Addison Wesley, 2001.

Ivan Bjerre Damgard. Practical and provably secure release of a secret and exchange of signatures. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 200–217, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc. ISBN 3-540-57600-2.

Yvo Desmedt. Some recent research aspects of threshold cryptography. In *ISW '97: Proceedings of the First International Workshop on Information Security*, pages 158–173, London, UK, 1998. Springer-Verlag. ISBN 3-540-64382-6. URL http://www.cs.fsu.edu/~desmedt/ISW97.pdf.

Leni Dharmawan. Dynamics of Local Capacity and Village Governance: Findings from the Second Indonesian Local Level Institutions Study. Central Java Report., 09 2002. URL http://siteresources.worldbank.org/INTINDONESIA/Resources/Social/Central+Java+Report+090802.pdf. Report for World Bank.

Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992. ISSN 0925-1022. doi: http://dx.doi.org/10.1007/BF00124891.

Carl Ellison and Bruce Schneier. Ten risks of PKI: What you're not being told about public-key infrastructure. *Computer Security Journal*, 16(1):1–7, ???? 2000. ISSN 0277-0865. URL http://www.schneier.com/paper-pki.pdf.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

Diego Gambetta. *Can We Trust Trust?*, chapter 13, pages 213–237. Basil Blackwell, 1988. URL citeseer.ifi.unizh.ch/gambetta88can.html. Reprinted in electronic edition from Department of Sociology, University of Oxford, chapter 13, pp. 213-237".

G. Nigel Gilbert and Klaus G. Troitzsch. *Simulation for the Social Scientist*. Taylor & Francis, Inc., Bristol, PA, USA, 1999. ISBN 0335197450.

S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. ISSN 0097-5397. doi: http://dx.doi.org/10.1137/0218012.

Tyrone Grandison and Morris Sloman. Sultan - a language for trust specification and analysis. In *he 8th Annual Workshop HP OpenView University Association (HP-OVUA)*, 2001.

Roslan Ismail and Audun Jøsang. The Beta Reputation System. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002.

A. Jones and M. Sergot. A formal characterization of institutionalized power. *Journal of the IGPL*, 4(3):429–445, 1996. URL http://citeseer.ist.psu.edu/jones96formal.html.

Audun Josang. Trust-based decision making for electronic transactions. In L. Yngstrin and T. Svensson, editors, *Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99)*. Stockholm University, Sweden, 1999.

Audun Jøsang and Stéphane Lo Presti. Analysing the relationship between risk and trust. In *Proceedings of Second International Conference on Trust Management (iTrust 2004) LNCS 2995*, pages 135–145, 2004.

Audun Jøsang, Roslan Ismail, and Colin Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 2005.

Audun Josang, Claudia Keser, and Theo Dimitrakos. Can we manage trust. In *Third International Conference, iTrust 2005, Paris, France*, 2005. URL http://sky.fit.qut.edu.au/~josang/papers/JKD2005-iTrust.pdf.

Loren M Kohnfelder. Towards a practical public-key cryptosystem. Bachelor Thesis of Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1978. URL http://dspace.mit.edu/handle/1721.1/15993.

Leslie Lamport. The TLA homepage, 11 2001. URL http://research.microsoft.com/users/lamport/tla/tla.html.

Leslie Lamport. *Specifying Systems*. Addison-Wesley, 2003.

Leslie Lamport. Tla in pictures. *IEEE Trans. Softw. Eng.*, 21(9):768–775, 1995. ISSN 0098-5589. doi: http://dx.doi.org/10.1109/32.464544.

Kenli Li, Yan He, Xiaoling Liu, and Ying Wang. Security-driven scheduling algorithms based on eigentrust in grid. In *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, pages 1068–1072, 05-08 Dec. 2005. doi: 10.1109/PDCAT.2005.212.

Ninghui Li and John C. Mitchell. Rt: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition DISCEX III*, 2003. URL http://crypto.stanford.edu/ninghui/papers/rt_discex03.pdf.

Zhengqiang Liang and Weisong Shi. Performance evaluation of rating aggregation algorithms in reputation systems. In *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on*, page 10pp., 19-21 Dec. 2005. doi: 10.1109/COLCOM.2005.1651235.

Nicolas Liebau, Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. A Token-based Accounting Scheme for P2P-Systems. Technical Report TR-2004-05, Technische Universitaet Darmstadt, January 2004. URL http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/LDMS04-1.html.

Nicolas Liebau, Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. Token-based Accounting for P2P-Systems. In *Proceeding of Kommunikation in Verteilten Systemen KiVS 2005*, pages 16–28, February 2005. URL http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/LDMS05-1.html. (Received Best Paper Award).

Niklas Luhmann. *Trust and Power*. Wiley, Chichester, 1979.

Michael Malkin, Thomas Wu, and Dan Boneh. Experimenting with Shared Generation of RSA keys. In *Proceedings of the Internet Society's 1999 Symposium on Network and Distributed System Security (SNDSS)*, 1999.

Daniel W. Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, page 312, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8292-2.

John C. Marchesini and Sean W. Smith. Virtual Hierarchies - An Architecture for Building and Maintaining Efficient and Resilient Trust Chains. Technical Report TR2002-416, Dartmouth College, Computer Science, Hanover, NH, February 2002. URL ftp://ftp.cs.dartmouth.edu/TR/TR2002-416.ps.Z.

Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, 1994.

T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *Transactions of the IECE (Japan)*, 69:99–106, 1986.

D. Harrison Mcknight and Norman L. Chervany. The meaning of trust. Technical report, Carlson School of Management, University of Minnesota, 1996.

Edward Miguel, Paul Gertler, and David I. Levine. Did Industrialization Destroy Social Capital in Indonesia?, 06 2005. URL http://faculty.haas.berkeley.edu/gertler/working_papers/sk-indonesia_07jun02.pdf. p.5,6,27.

Barbara A. Misztal. *Trust in Modern Societies*. Polity Press, 1996.

Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/359657.359659.

Aleksandra Nenadic. *A Security Solution for Fair Exchange and Non-Repudiation in E-Commerce*. PhD thesis, Faculty of Engineering and Physical Sciences, University of Manchester, July 2005. URL http://www.cs.man.ac.uk/~nenadic/publications/Thesis.pdf.

Tatsuaki Okamoto and Kazuo Ohta. How to simultaneously exchange secrets by general assumptions. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 184–192, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-732-4. doi: http://doi.acm.org/10.1145/191177.191221.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. URL citeseer.nj.nec.com/page98pagerank.html.

Daniele Quercia, Stephen Hailes, and Licia Capra. Tata: Towards anonymous trusted authentication. In *The 4th International Conference on Trust Management (ITrust)*, 2004. URL http://www.cs.ucl.ac.uk/staff/l.capra/publications/querciaTATA06.pdf.

Jean-Jacques Quisquater, Louis Guillou, Marie Annick, and Tom Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 628–631, New York, NY, USA, 1989. Springer-Verlag New York, Inc. ISBN 0-387-97317-6.

Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce*, volume 11 of Advances in Applied Microeconomics. Elsevier Science, 2002.

Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-326-X. doi: http://doi.acm.org/10.1145/375735.376110.

Andreas Schlosser, Marco Voss, and Lars Bruckner. Comparing and evaluating metrics for reputation systems by simulation. *A Workshop on Reputation in Agent Societies as part of 2004 IEEE/WIC/ACM International Joint Conference on Intelligent Agent*, 2004. URL http://www.ito.tu-darmstadt.de/publs/pdf/pdf/SchlosserEtAl-ComparingReputationMetricsBySimulation-RAS2004.pdf.

John R. Searle. *The Construction of Social Reality*. The Free Press, 1995.

John R. Searle. Social ontology and political power, 10 2005a. URL http://www.law.berkeley.edu/centers/kadish/searle.pdf.

John R. Searle. Social ontology: Some basic principles, 10 2005b. URL http://ist-socrates.berkeley.edu/~jsearle/AnthropologicalTheoryFNLversion.doc.

Hector Garcia-Molina Sepandar D. Kamvar, Mario T. Schlosser. The eigentrust algorithm for reputation management in p2p networks. In *In Proceedings of the Twelfth International World Wide Web Conference*, 2003.

Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/359168.359176.

Barry Smith. From speech acts to social reality, 10 2005. URL http://ontology.buffalo.edu/smith/articles/SearleIntro.pdf.

Barry Smith and John Searle. The construction of social reality: An exchange, 10 2006. URL http://ontology.buffalo.edu/smith/articles/dksearle.htm.

The Internet Society. Diffie-hellman key agreement method. http://tools.ietf.org/html/rfc2631, June 1999. URL http://tools.ietf.org/html/rfc2631.

Frank Stajano and Ross J. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *IH '99: Proceedings of the Third International Workshop on Information Hiding*, pages 434–447, London, UK, 2000. Springer-Verlag. ISBN 3-540-67182-X.

Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In USENIX Association, editor, *USENIX Conference Proceedings (Dallas, TX, USA)*, pages 191–202, Berkeley, CA, USA, Winter 1988. USENIX Association. URL http://www.cs.utah.edu/~{}retrac/cs6961/kerberos88.ps.gz.

Piotr Stompzka. *Trust: A Sociological Theory*. Cambridge University Press, 1999.

Avinanta Tarigan. Towards communally governed transactions among decentralized trading agents. In *Second International IEEE Workshop on the Value of Security through Collaboration (SECOVAL 2006)*, 2006.

Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gun Sirer. Karma: A secure economic framework for p2p resource sharing. In *Workshop on the Economics of Peer-to-Peer Systems*, 2003. URL http://www.cs.cornell.edu/people/egs/papers/karma.pdf.

Merriam Webster. Merriam-webster online dictionary on definition of transaction, 08 2005. URL http://www.webster.com/cgi-bin/dictionary/transaction.

Wikipedia. Financial transaction, 09 2005. URL http://en.wikipedia.org/wiki/Financial_transaction.

Uri Wilensky. Netlogo itself. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999. URL http://ccl.northwestern.edu/netlogo/.

Uri Wilensky and Walter M. Stroup. Hubnet. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999. URL http://ccl.northwestern.edu/netlogo/hubnet.html.

Walt Yao. Fidelis: A policy-driven trust management framework. In *iTrust*, Lecture Notes in Computer Science, pages 301–317. Springer, 2003. URL http://link.springer.de/link/service/series/0558/bibs/2692/26920301.htm.

Bin Yu and Munindar P. Singh. Incentive mechanisms for peer-to-peer systems. *Agents and Peer-to-Peer Computing, Second International Workshop*, pages 77–88, 2003. URL http://citeseer.ist.psu.edu/647427.html.

Appendix A

Transaction Algorithm in TLA+

MODULE *CGTSystem*

Specification / abstraction of the algorithm of

Communal Governed Transactions Among Decentralized Trading Agents

Avinanta *Tarigan*

AG Rechnernetze und verteilte *Systeme*

Technische *Fakultaet – Universitaet Bielefeld*

Last Updated : 8 *June* 2007

This specification describes governed transaction by distributed and decentralized trading agents where no mediator or authority exists to facilitate transactions. System behaviour which emerge through individual action are the accountability of so-called institutional-facts and the communal authorization for proposed transaction. Individual trust decision which emerges communal authorization is based on the rational trust decision (Coleman) and trust information is calculated from propagated binary report from each conducted transaction using Beta algorithm. Assumption taken in this specification is that the communication channel is reliable.

EXTENDS

Naturals, *Reals*, *Sequences*, *FiniteSets*

CONSTANTS

community,	Set of agents
m,	Majority Threshold $0.5 \leq m \leq 1$
p_assertions,	Set of Possible assertions to be proposed
p_messages,	Set of Possible Exchanges Messages
init_ifacts,	Initial Institutional Facts
propose,	Identifier for message containing proposal
statement,	Identifier for message containing statement
approve,	Identifier for approving proposal
noapprove,	Identifier for not approving proposal
rating,	Identifier for message containing rating
good,	Identifier for good results
bad,	Identifier for bad result

p_proposals, Set of possible proposal
init_good_exp, Initial good experience
init_bad_exp Initial bad experience

VARIABLES

agent, Agents variables
buyer, *seller*, One of buyer and *seller* ∈ *community* used to create encounter
lgratio, Loss to Gain Ratio
P_o, Set of instituional *facts* ∈ *p_assertions* to be eliminated
P_n Set of assertions to be proposed as institutional-facts

Definition of agent's variable which holds internal memory of agents

$$
\begin{array}{l}
TypeAgent \triangleq \\
\quad \land\ agent \in [community \\
\qquad \to [ifacts \ : \{f \in p\_assertions : \text{TRUE}\}, \\
\qquad\ \ msgfifo \ : Seq(p\_messages), \\
\qquad\ \ experience : [community \\
\qquad\qquad \to [sum\_good \ : Nat, \\
\qquad\qquad\ \ sum\_bad \ : Nat \ ]], \\
\qquad\ \ proposals \ : [p\_proposals \\
\qquad\qquad \to [proposed \ : \text{BOOLEAN}\ , \\
\qquad\qquad\ \ decided \ : \text{BOOLEAN}\ , \\
\qquad\qquad\ \ rated \ : \text{BOOLEAN}\ , \\
\qquad\qquad\ \ sum\_approved : Nat, \\
\qquad\qquad\ \ sum\_notapproved : Nat]] \\
\qquad] \\
\quad]
\end{array}
$$

Variables Definition

$$
\begin{array}{l}
TypeInvariant \triangleq \\
\quad \land\ buyer \in community \\
\quad \land\ seller \in community \\
\quad \land\ buyer \neq seller \\
\quad \land\ m \in Real \\
\quad \land\ P_o \ \in \text{SUBSET}\ p\_assertions \\
\quad \land\ P_n \ \in \text{SUBSET}\ p\_assertions \\
\quad \land\ TypeAgent
\end{array}
$$

Variabel initialization

$$
\begin{array}{l}
\mathit{InitAgent} \triangleq \\
\quad \land\ \mathit{agent} \in [\mathit{community} \\
\qquad\qquad \rightarrow [\mathit{ifacts} \quad : \mathit{init\_ifacts}, \\
\qquad\qquad\quad \mathit{msgfifo} \quad : \langle\rangle, \\
\qquad\qquad\quad \mathit{experience} \ : [\mathit{community} \\
\qquad\qquad\qquad\qquad\quad \rightarrow [\mathit{sum\_good} \ : \mathit{init\_good\_exp}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_bad} \ : \mathit{init\_bad\_exp}\ ]], \\
\qquad\qquad\quad \mathit{proposals} \ : [\mathit{p\_proposals} \\
\qquad\qquad\qquad\qquad\quad \rightarrow [\mathit{proposed} \quad : \text{FALSE}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{decided} \quad : \text{FALSE}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{rated} \quad : \text{FALSE}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_approved} : 0, \\
\qquad\qquad\qquad\qquad\qquad \mathit{sum\_notapproved} : 0]] \\
\qquad\qquad\quad] \\
\qquad\qquad]
\end{array}
$$

Function to construct record of a message to be sent

$$
\begin{array}{l}
\mathit{msgStruct} \triangleq \\
\quad [\mathit{thesenders} \in \text{SUBSET } \mathit{community}, \\
\quad\ \mathit{thebody} \quad \in \mathit{p\_messages} \\
\qquad\qquad \mapsto [\mathit{senders} : \mathit{thesenders}, \\
\qquad\qquad\quad \mathit{body} \quad : \mathit{thebody} \quad] \in \mathit{p\_messages} \\
\quad]
\end{array}
$$

Function to simplify the construction of a proposal

$$
\begin{array}{l}
\mathit{propStruct} \triangleq \\
\quad [\mathit{thebuyer}, \mathit{theseller} \in \mathit{community}, \\
\quad\ \mathit{o\_facts}, \mathit{n\_asserts} \ \in \text{SUBSET } \mathit{p\_assertions}, \\
\quad\ \mathit{lg} \in \mathit{Real} \\
\qquad\qquad \mapsto [\mathit{id} \quad : \mathit{propose}, \\
\qquad\qquad\quad \mathit{buyer} \quad : \mathit{thebuyer}, \\
\qquad\qquad\quad \mathit{seller} \quad : \mathit{theseller}, \\
\qquad\qquad\quad P_o \quad : \mathit{o\_facts}, \\
\qquad\qquad\quad P_n \quad : \mathit{n\_asserts}, \\
\qquad\qquad\quad \mathit{lgratio} \quad : \mathit{lg}] \in \mathit{p\_proposals} \\
\quad]
\end{array}
$$

Function to simplify the construction of a statement

$$
\mathit{statStruct} \triangleq
$$

```
  [astatus      ∈ {approve, noapprove},
   theproposal  ∈ p_proposals
                ↦ [id       : statement,
                   status   : astatus,
                   proposal : theproposal]
  ]
```

Function to simplify the construction of a report

```
repStruct ≜
  [π ∈ p_proposals,
   r  ∈ {good, bad}
              ↦ [id       : rating,
                 proposal : π,
                 result   : r  ]
  ]
```

Function to check if a proposal valid

```
isPropValid ≜
  [π     ∈ p_proposals,
   facts ∈ SUBSET p_assertions
           ↦ IF  ∧ π.buyer      ∈ community
                 ∧ π.seller     ∈ community
                 ∧ π.P_o        ∈ SUBSET facts
                 ∧ π.P_n        ∈ SUBSET p_assertions
             THEN TRUE
             ELSE FALSE
  ]
```

An Example of Action in Broadcasting a Message *msg* from subset of agents in senders

```
Broadcast(senders, msg) ≜
  ∧ senders ∈ SUBSET community
  ∧ msg     ∈ p_messages
  ∧ agent' = [agent EXCEPT ![∀ s ∈ community : s ∉ senders] =
                  [@ EXCEPT !.msgfifo =
                     Append(@.msgfifo, msgStruct[senders, msg])]]
```

Rational trust decision based on *Coleman*
The probability ρ is derived from experience of
the agents towards particular selling agent
which in this specification is computed using Beta reputation

```
trustDecision ≜
  [lg          ∈ Real,
   sum_good    ∈ Nat,
   sum_bad     ∈ Nat
       ↦ LET ρ ≜
                (sum_good + 1)/(sum_good + sum_bad + 2)
         IN    (IF ρ > (1 − ρ) ∗ lg
                THEN TRUE
                ELSE FALSE)
  ]
```

Agent actions in receiving a proposal
It begins by checking whether a message in the communication stack.
If the message received is a proposal then agent check the proposal
and decides whether or not to approve it, marks the proposal in
its knowledge, broadcast its statement to others, and wait for
other's statement

ReceiveProposalApprove determines approval of an agent on a proposal p
It adds its own statement as well as that of proposing agents (3)
in *sum_approved*
It is assumed that by proposing proposals, proposing agents already
approve their proposal

```
ReceiveProposalApprove ≜
  ∃ a ∈ community :
    ( ∧ Len(agent[a].msgfifo) > 0
      ∧ Head(agent[a].msgfifo).body.id = propose
      ∧ LET π ≜ Head(agent[a].msgfifo).body    IN
        ( ∧ isPropValid[π, agent[a].ifacts]
          ∧ trustDecision[π.lgratio,
                          agent[a].experience[π.seller].sum_good,
                          agent[a].experience[π.seller].sum_bad]
          ∧ agent′ = [agent EXCEPT
                        ![a] = [@ EXCEPT
                                  !.proposals[π].sum_approved =
                                    @.proposals[π].sum_approved + 3,
                                  !.proposals[π].proposed = TRUE,
                                  !.proposals[π].decided  = FALSE,
```

```
                    !.proposals[π].rated     = FALSE,
                    !.msgfifo = Tail(@.msgfifo)          ],
                ![∀ r ∈ community : r ≠ a] =
                    [@ EXCEPT
                      !.msgfifo = Append(@.msgfifo,
                          msgStruct[{a}, statStruct[approve, π]])]]
  ))
```

ReceiveProposalRefuse determines refusal of an agent on a proposal p
It adds its own statement on *sum_notapproved* (1)
and adds statement of the two proposing agents to *sum_approved* (2)
It is assumed that by proposing proposals, proposing agents already
approve their proposal

```
ReceiveProposalRefuse ≜
  ∃ a ∈ community :
    ( ∧ Len(agent[a].msgfifo) > 0
      ∧ Head(agent[a].msgfifo).body.id = propose
      ∧ LET π ≜ Head(agent[a].msgfifo).body    IN
        (( ∨ isPropValid[π, agent[a].ifacts] = FALSE
           ∨ trustDecision[π.lgratio,
                           agent[a].experience[π.seller].sum_good,
                           agent[a].experience[π.seller].sum_bad] = FALSE)
          ∧ agent' = [agent EXCEPT
                      ![a] = [@ EXCEPT
                                !.proposals[π].sum_approved =
                                   @.proposals[π].sum_approved + 2,
                                !.proposals[π].sum_approved =
                                   @.proposals[π].sum_notapproved + 1,
                                !.proposals[π].proposed = TRUE,
                                !.proposals[π].decided  = FALSE,
                                !.proposals[π].rated    = FALSE,
                                !.msgfifo = Tail(@.msgfifo)],
                      ![∀ r ∈ community : r ≠ a] =
                          [@ EXCEPT
                            !.msgfifo = Append(@.msgfifo,
                                msgStruct[{a}, statStruct[noapprove, π]])]]))
```

ReceiveStatementApproval determines action that agent does on receiving
a statement from other regarding the approval of proposal p
This particular agent adds one count of the approval of p in *.sum_approved*

$$
\begin{array}{l}
\mathit{ReceiveStatementApproval} \triangleq \\
\quad \exists\, a \in \mathit{community} : \\
\quad\quad (\ \wedge \mathit{Len}(\mathit{agent}[a].\mathit{msgfifo}) > 0 \\
\quad\quad\ \ \wedge \mathit{Head}(\mathit{agent}[a].\mathit{msgfifo}).\mathit{body}.\mathit{id} = \mathit{statement} \\
\quad\quad\ \ \wedge \text{LET } s \triangleq \mathit{Head}(\mathit{agent}[a].\mathit{msgfifo}).\mathit{body} \text{ IN} \\
\quad\quad\quad\quad (\ \wedge \mathit{agent}[a].\mathit{proposals}[s.\mathit{proposal}].\mathit{proposed} \\
\quad\quad\quad\quad\ \ \wedge s.\mathit{status} = \mathit{approve} \\
\quad\quad\quad\quad\ \ \wedge \mathit{agent}' = [\mathit{agent} \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad ![a] = [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.\mathit{proposals}[s.\mathit{proposal}].\mathit{sum\_approved} = \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad @.\mathit{proposals}[s.\mathit{proposal}].\mathit{sum\_approved} + 1, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.\mathit{msgfifo} = \mathit{Tail}(@.\mathit{msgfifo})]] \\
\quad\quad\quad\quad) \\
\quad\quad)
\end{array}
$$

ReceiveStatementApproval determines action that agent does on receiving
a statement from other regarding the refusal of proposal p
This particular agent adds one count of the refusal of p in *.sum_notapproved*

$$
\begin{array}{l}
\mathit{ReceiveStatementRefusal} \triangleq \\
\quad \exists\, a \in \mathit{community} : \\
\quad\quad (\ \wedge \mathit{Len}(\mathit{agent}[a].\mathit{msgfifo}) > 0 \\
\quad\quad\ \ \wedge \mathit{Head}(\mathit{agent}[a].\mathit{msgfifo}).\mathit{body}.\mathit{id} = \mathit{statement} \\
\quad\quad\ \ \wedge \text{LET } s \triangleq \mathit{Head}(\mathit{agent}[a].\mathit{msgfifo}).\mathit{body} \text{ IN} \\
\quad\quad\quad\quad (\ \wedge \mathit{agent}[a].\mathit{proposals}[s.\mathit{proposal}].\mathit{proposed} \\
\quad\quad\quad\quad\ \ \wedge s.\mathit{status} \neq \mathit{approve} \\
\quad\quad\quad\quad\ \ \wedge \mathit{agent}' = [\mathit{agent} \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad ![a] = [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.\mathit{proposals}[s.\mathit{proposal}].\mathit{sum\_notapproved} = \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad @.\mathit{proposals}[s.\mathit{proposal}].\mathit{sum\_notapproved} + 1, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.\mathit{msgfifo} = \mathit{Tail}(@.\mathit{msgfifo})]] \\
\quad\quad\quad\quad) \\
\quad\quad)
\end{array}
$$

ReceiveStatementUndetermined determines action that agent does on receiving
a statement from other but the proposal is not yet received by particular agent
This particular agent moves the message to the end of *msgfifo* while waiting
for the proposal itself
This is precaution of unsyncronization of communication channel

$$
\mathit{ReceiveStatementUndetermined} \triangleq
$$

$$
\begin{array}{l}
\exists\, a \in community : \\
\quad (\, \wedge Len(agent[a].msgfifo) > 0 \\
\quad\ \wedge Head(agent[a].msgfifo).body.id = statement \\
\quad\ \wedge \text{LET } s \triangleq Head(agent[a].msgfifo).body \ \text{ IN} \\
\qquad (\, \wedge agent[a].proposals[s.proposal].proposed = \text{TRUE} \\
\qquad\ \wedge agent' = [agent \ \text{EXCEPT} \\
\qquad\qquad\qquad ![a] = [@ \ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad !.msgfifo = \\
\qquad\qquad\qquad\qquad Tail(@.msgfifo) \circ \langle Head(@.msgfifo)\rangle]] \\
\qquad) \\
\quad\ )
\end{array}
$$

ReceiveRatingGood determines action that agent does on receiving a "Good" rating from transacting agent regarding past approved transaction This particular agent adds count the rating for seller agent in *experience*[*selleragent*].*sum_good*

$$
\begin{array}{l}
ReceiveRatingGood \triangleq \\
\quad \exists\, a \in community : \\
\quad\ (\, \wedge Len(Head(agent[a].msgfifo)) > 0 \\
\quad\ \ \wedge Head(agent[a].msgfifo).body.id = rating \\
\quad\ \ \wedge \text{LET } msg \triangleq Head(agent[a].msgfifo.body) \text{IN} \\
\qquad (\, \wedge agent[a].proposals[msg.proposal].rated \ \ = \text{FALSE} \\
\qquad\ \ \wedge agent[a].proposals[msg.proposal].decided \ = \text{TRUE} \\
\qquad\ \ \wedge agent[a].proposals[msg.proposal].proposed = \text{FALSE} \\
\qquad\ \ \wedge msg.result = good \\
\qquad\ \ \wedge agent' = [agent \ \text{EXCEPT} \ ![a] = [@ \ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad !.proposals[msg.proposal].rated = \text{TRUE}, \\
\qquad\qquad\qquad\qquad !.experience[msg.proposal.seller].sum\_good = \\
\qquad\qquad\qquad\qquad\ @.experience[msg.proposal.seller].sum\_good + 1, \\
\qquad\qquad\qquad\qquad !.msgfifo = Tail(@.msgfifo)]] \\
\qquad) \\
\quad\ )
\end{array}
$$

ReceiveRatingBad determines action that agent does on receiving a "Bad" rating from transacting agent regarding past approved transaction This particular agent adds count the rating for seller agent in *experience*[*selleragent*].*sum_bad*

$$
\begin{array}{l}
ReceiveRatingBad \triangleq \\
\quad \exists\, a \in community :
\end{array}
$$

$$
\begin{array}{l}
(\ \wedge\ Len(Head(agent[a].msgfifo)) > 0 \\
\quad \wedge\ Head(agent[a].msgfifo).body.id = rating \\
\quad \wedge\ \text{LET}\ msg \triangleq Head(agent[a].msgfifo.body)\text{IN} \\
\qquad (\ \wedge\ agent[a].proposals[msg.proposal].rated \quad = \text{FALSE} \\
\qquad\ \wedge\ agent[a].proposals[msg.proposal].decided \ = \text{TRUE} \\
\qquad\ \wedge\ agent[a].proposals[msg.proposal].proposed = \text{FALSE} \\
\qquad\ \wedge\ agent' = [agent\ \text{EXCEPT}\ ![a] = [@\ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad !.proposals[msg.proposal].rated = \text{TRUE}, \\
\qquad\qquad\qquad\qquad !.experience[msg.proposal.seller].sum\_bad = \\
\qquad\qquad\qquad\qquad\ @.experience[msg.proposal.seller].sum\_bad + 1, \\
\qquad\qquad\qquad\qquad !.msgfifo = Tail(@.msgfifo)]] \\
\qquad) \\
\quad)
\end{array}
$$

ReceiveRatingUndetemined determines action that agent does on receiving a rating from other but the corresponding proposal does not receive as approved by community yet

This particular agent moves the message to the end of *msgfifo* while waiting for the proposal itself to be approved

This is precaution of unsyncronization of communication channel

$$
\begin{array}{l}
ReceiveRatingUndetermined \triangleq \\
\exists\, a \in community : \\
\quad (\ \wedge\ Len(Head(agent[a].msgfifo)) > 0 \\
\quad\ \wedge\ Head(agent[a].msgfifo).body.id = rating \\
\quad\ \wedge\ \text{LET}\ msg \triangleq Head(agent[a].msgfifo.body)\text{IN} \\
\qquad ((\ \vee\ agent[a].proposals[msg.proposal].rated \quad = \text{TRUE} \\
\qquad\ \vee\ agent[a].proposals[msg.proposal].decided \ = \text{FALSE} \\
\qquad\ \vee\ agent[a].proposals[msg.proposal].proposed = \text{TRUE}) \\
\qquad\ \wedge\ agent' = [agent\ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad ![a] = [@\ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad\qquad\qquad !.msgfifo = \\
\qquad\qquad\qquad\qquad\qquad\qquad\ Tail(@.msgfifo) \circ \langle Head(@.msgfifo)\rangle]] \\
\qquad) \\
\quad)
\end{array}
$$

ProcCommunityApproval determines action that agent does when the proposal p is considered to be approved by community

ifacts is the list of actual institutional facts

operation $ifacts' = ifacts - \pi.P_o + \pi.P_n$ is what agent does to accept each new assertions in $\pi.P_n$

and to reject each institutional facts in $\pi.P_o$

$$
\begin{array}{l}
ProcCommunityApproval \triangleq \\
\quad \exists\, a,\, \pi \in community \times p\_proposals : \\
\quad (\, \wedge\, agent[a].proposals[\pi].proposed = \text{TRUE} \\
\quad\ \wedge\, agent[a].proposals[\pi].decided = \text{FALSE} \\
\quad\ \wedge\, agent[a].proposals[\pi].sum\_approved + \\
\qquad\quad agent[a].proposals[\pi].sum\_notapproved = Cardinality(community) \\
\quad\ \wedge\, agent[a].proposals[\pi].sum\_approved \geq (Cardinality(community) * m) \\
\quad\ \wedge\, agent' = [agent \text{ EXCEPT} \\
\qquad\qquad\qquad ![a] = [@ \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad !.ifacts = \\
\qquad\qquad\qquad\qquad\quad \text{UNION } \{(@.ifacts \setminus \pi.P_o),\, \pi.P_n\}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].proposed = \text{FALSE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].decided \ \ = \text{TRUE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].sum\_approved = 0, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].sum\_notapproved = 0]])
\end{array}
$$

ProcCommunityRefusal determines action that agent does when the proposal p is considered to be refused by community

$$
\begin{array}{l}
ProcCommunityRefusal \triangleq \\
\quad \exists\, a,\, \pi \in community \times p\_proposals : \\
\quad (\, \wedge\, agent[a].proposals[\pi].proposed = \text{TRUE} \\
\quad\ \wedge\, agent[a].proposals[\pi].decided = \text{FALSE} \\
\quad\ \wedge\, agent[a].proposals[\pi].sum\_approved + \\
\qquad\quad agent[a].proposals[\pi].sum\_notapproved = Cardinality(community) \\
\quad\ \wedge\, agent[a].proposals[\pi].sum\_approved < Cardinality(community) * m \\
\quad\ \wedge\, agent' = [agent \text{ EXCEPT} \\
\qquad\qquad\qquad ![a] = [@ \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad !.proposals[\pi].rated = \text{FALSE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].proposed = \text{FALSE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].decided \ \ = \text{FALSE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].sum\_approved = 0, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].sum\_notapproved = 0]])
\end{array}
$$

rateTransResult determines a judgement of agent of approved proposal (or transaction) p

$$
\begin{array}{l}
rateTransResult \triangleq \\
\quad [\pi \in p\_proposals \\
\qquad \mapsto \text{CHOOSE } x \in \{bad,\, good\} : \text{TRUE}]
\end{array}
$$

ProcRatingGood determines action that buying agent does to rate approved *proposal*/ transaction as GOOD
It broadcast to all agent about his opinion /*rating*

$$
\begin{array}{l}
ProcRatingGood \triangleq \\
\quad \exists\, a,\, \pi \in community \times p\_proposals : \\
\quad\quad (\, \wedge\, agent[a].proposals[\pi].decided = \text{TRUE} \\
\quad\quad\ \wedge\, agent[a].proposals[\pi].rated = \text{FALSE} \\
\quad\quad\ \wedge\, \pi.buyer = a \\
\quad\quad\ \wedge\, rateTransResult[\pi] = good \\
\quad\quad\ \wedge\, agent' = [agent \\
\quad\quad\quad\quad\quad\quad \text{EXCEPT } ![\forall\, r \in community : r \neq a] = \\
\quad\quad\quad\quad\quad\quad\quad\quad [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad !.msgfifo = Append(@.msgfifo, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad msgStruct[\{a\},\, repStruct[\pi,\, good]])], \\
\quad\quad\quad\quad\quad\quad\quad\quad ![a] = \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.proposals[\pi].rated = \text{TRUE}, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.experience[\pi.seller].sum\_good = \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad @.experience[\pi.seller].sum\_good + 1]] \\
\quad\quad)
\end{array}
$$

ProcRatingBad determines action that buying agent does to rate approved *proposal*/ transaction as BAD
It broadcast to all agent about his opinion /*rating*

$$
\begin{array}{l}
ProcRatingBad \triangleq \\
\quad \exists\, a,\, \pi \in community \times p\_proposals : \\
\quad\quad (\, \wedge\, agent[a].proposals[\pi].decided = \text{TRUE} \\
\quad\quad\ \wedge\, agent[a].proposals[\pi].rated = \text{FALSE} \\
\quad\quad\ \wedge\, \pi.buyer = a \\
\quad\quad\ \wedge\, rateTransResult[\pi] = bad \\
\quad\quad\ \wedge\, agent' = [agent \\
\quad\quad\quad\quad\quad\quad \text{EXCEPT } ![\forall\, r \in community : r \neq a] = \\
\quad\quad\quad\quad\quad\quad\quad\quad [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad !.msgfifo = Append(@.msgfifo, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad msgStruct[\{a\},\, repStruct[\pi,\, bad]])], \\
\quad\quad\quad\quad\quad\quad\quad\quad ![a] = \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad [@ \text{ EXCEPT} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.proposals[\pi].rated = \text{TRUE}, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad !.experience[\pi.seller].sum\_bad =
\end{array}
$$

$$@.experience[\pi.seller].sum\_bad + 1]]$$

)

Propose determines action from proposing agents (thbuyer and *theseller*) to :
1. Construct the proposal p
2. Broadcast the proposal p to all user except them
3. Mark the proposal p in *thebuyer*'s and *theseller*'s internal knowledges
as approved by 2 agents (themself)

$$
\begin{array}{l}
Propose(thebuyer,\ theseller,\ O,\ N,\ lg) \triangleq \\
\quad \land\ thebuyer \in community \\
\quad \land\ theseller \in community \\
\quad \land \text{LET } \pi \triangleq propStruct[thebuyer,\ theseller,\ O,\ N,\ lg] \text{IN} \\
\qquad (\land\ agent' = [agent \\
\qquad\qquad \text{EXCEPT } ![\forall\ a \in community : a \notin \{thebuyer,\ theseller\}] = \\
\qquad\qquad\qquad [@ \text{ EXCEPT} \\
\qquad\qquad\qquad\quad !.msgfifo = \\
\qquad\qquad\qquad\qquad Append(@.msgfifo,\ msgStruct[\{thebuyer,\ theseller\},\ \pi])], \\
\qquad\qquad\qquad ![\{thebuyer,\ theseller\}] = \\
\qquad\qquad\qquad\quad [@ \text{ EXCEPT} \\
\qquad\qquad\qquad\qquad !.proposals[\pi].sum\_approved = \\
\qquad\qquad\qquad\qquad\qquad @.proposals[\pi].sum\_approved + 2, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].proposed = \text{TRUE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].decided\ = \text{FALSE}, \\
\qquad\qquad\qquad\qquad !.proposals[\pi].rated\ \ = \text{FALSE}]])
\end{array}
$$

Encounter is the event in which two agents meet and want to conduct
transaction by proposing new assertion which will override
their old institutional-facts when they are accepted by
community

$$
\begin{array}{l}
Encounter \triangleq \\
\quad \land \langle buyer,\ seller \rangle \\
\qquad\quad [\text{CHOOSE } \langle x,\ y \rangle \in community^2 : x \neq y] \\
\quad \land \langle P_o,\ P_n \rangle \\
\qquad\quad [\text{CHOOSE } \langle O,\ N \rangle \in (\text{SUBSET } p\_assertions)^2 : \\
\qquad\qquad\quad \land\ O \subseteq agent[buyer].ifacts \\
\qquad\qquad\quad \land \forall\ x \in O : x \notin N \\
\qquad\qquad\quad \land \forall\ x \in N : x \notin O\ ] \\
\quad \land\ lgratio\ [\text{CHOOSE } x \in Real : x > 0]
\end{array}
$$

$\land\ Propose(buyer,\ seller,\ P_o,\ P_n,\ lgratio)$

Next is action predicate that "always" enabled /TRUE
due to □ always operator
In a state of a *Next*, there might be an action predicate
which is enabled or TRUE according to the pre-condition
The action changes the system variable agent which represents
composition of internal knowledge of all agents

$$
\begin{aligned}
Next \triangleq\ & \lor Encounter \\
& \lor ReceiveProposalApprove \\
& \lor ReceiveProposalRefuse \\
& \lor ReceiveStatementApproval \\
& \lor ReceiveStatementRefusal \\
& \lor ReceiveStatementUndetermined \\
& \lor ProcCommunityApproval \\
& \lor ProcCommunityRefusal \\
& \lor ProcRatingGood \\
& \lor ProcRatingBad \\
& \lor ReceiveRatingGood \\
& \lor ReceiveRatingBad \\
& \lor ReceiveRatingUndetermined
\end{aligned}
$$

$Spec \triangleq InitAgent \land \Box[Next]_{agent}$

THEOREM $Spec \implies \Box TypeInvariant$

Appendix B

Source of Simulation Program

```
;; Defining Local Agents Variables
turtles-own [
    message?
    behaviourid    ;; Determine behaviour of agent
    badopinions    ;; Beta Rep
    goodopinions   ;; Beta Rep
    fitness        ;; Fitness of agent
    rangeoutcome   ;; Possible range of outcome based on predefined behavior
    theID          ;; ID
    seller-turtle  ;; Variable to store the id of seller turtle
    diestatus   ;; Determine the life state
    opinions    ;; Avg Rep and Beta Rep
    dumpforeach ;; Internal purpose
    encountered ;; To determine whether an agent has transaction
    falseNegative ;; To Deterime falseNegative
    falsePositive ;; To Deterime falsePositive
    asSeller      ;; To Deterime Seller

;; Defining Global Variabels
]
globals [
    totalagents
    tick
    tagents
    listagents
    runnumber
    diehonest
    diefair
    dieunpredictable
    diedishonest
    minturtle
    ex1
    ex2
    ex3
    ex4
    ecRate

]

;; ---------- START SETUP -----------------
to setup
  ct
  cp
  clear-all-plots
  set ecRate 0
  set totalagents Honest + Dishonest + Fair + Unpredictable
  set listagents n-values totalagents [ ? + 0 ]
```

```
set tick 0
set tagents 0
set minturtle 5
set diehonest 300000
set diefair 300000
set dieunpredictable 300000
set diedishonest 300000

;; Begin Creating the Agents

cct Honest [ ;; Honest Agents behaviourid 1
  set diestatus false
  set theID tagents
  set tagents tagents + 1
  set message? false
  setxy random-xcor random-ycor
  set behaviourid 1
  set color blue
  set fitness InitialFitness
  set goodopinions n-values totalagents [ InitialReputation * 5 ]
  set badopinions n-values totalagents [ ( 1 - InitialReputation ) * 5 ]
  set rangeoutcome n-values 31 [ ? + 70 ]
  set encountered False
  set falsePositive False
  set falseNegative False
  set asSeller False

]
cct Fair [ ;; Fair Agents behaviourid  2
  set diestatus false
  set theID tagents
  set tagents tagents + 1
  set message? false
  setxy random-xcor random-ycor
  set behaviourid 2
  set color green
  set fitness InitialFitness
  set goodopinions n-values totalagents [ InitialReputation * 5 ]
  set badopinions n-values totalagents [ ( 1 - InitialReputation ) * 5 ]
  set rangeoutcome n-values 31 [ ? + 40 ]
  set encountered False
  set falsePositive False
  set falseNegative False
  set asSeller False


]
cct Unpredictable [ ;; Unpredictable Agents behaviourid 3
  set diestatus false
  set theID tagents
  set tagents tagents + 1
  set message? false
  setxy random-xcor random-ycor
  set behaviourid 3
  set color brown
  set fitness InitialFitness
  set goodopinions n-values totalagents [ InitialReputation * 5 ]
  set badopinions n-values totalagents [ ( 1 - InitialReputation ) * 5 ]
  set rangeoutcome n-values 100 [ ? + 0 ]
  set encountered False
  set falsePositive False
  set falseNegative False
  set asSeller False


]
cct Dishonest [ ;; DisHonest Agents behaviourid 4
  set diestatus false
```

```
    set theID tagents
    set tagents tagents + 1
    set message? false
    setxy random-xcor random-ycor
    set behaviourid 4
    set color red
    set fitness InitialFitness
    set goodopinions n-values totalagents [ InitialReputation * 5 ]
    set badopinions n-values totalagents [ ( 1 - InitialReputation ) * 5 ]
    set rangeoutcome n-values 40 [ ? + 1 ]
    set encountered False
    set falsePositive False
    set falseNegative False
    set asSeller False

  ]
  ask turtles [
    set opinions n-values totalagents [ 0 ]
    ifelse RepMethod = "AVERAGE" [
       ifelse RandomInitialRep = True [
          set opinions n-values totalagents [ random-float(1) ]
       ][
          set opinions n-values totalagents [ InitialReputation ]
       ]
    ][
      set dumpforeach n-values totalagents [ ? + 0 ]
      without-interruption [
      if RandomInitialRep = True [
         foreach n-values totalagents [ ? ] [
           set goodopinions replace-item ? goodopinions random 2
           set badopinions replace-item ? badopinions random 2  ]]
         foreach n-values totalagents [ ? ] [
           let i ?
           let g item i goodopinions
           let b item i badopinions
           let P ( ( g + 1 ) / ( g + b + 2 ) )
           set opinions replace-item i opinions P ]
      ]
    set size 1
    ]]
  set ex1 one-of turtles with [ behaviourid = 1 ]
  set ex2 one-of turtles with [ behaviourid = 2 ]
  set ex3 one-of turtles with [ behaviourid = 3 ]
  set ex4 one-of turtles with [ behaviourid = 4 ]
end
;; ------------ END SETUP --------------------

;; ------------ START LOOP -------------------
;; loop the simulation process
to goSimulate
  no-display
  set tick tick + 1
  every 0.1 [
      ask turtles [ moverandomly ]  ;; Ask turtles to move randomly
      ask turtles [
             set encountered False
             set falsePositive False
             set falseNegative False
             set asSeller False
             if diestatus = false
               [  ifelse DecMethod = "RECOMMENDATION"
                     [ tranRecommendation ]
                     [ tranVote ]
               ]
             if fitness < 1 [ set diestatus true ] ;; Ask turtles to die if lacks of fitness
    ]
  ]
  if PlotPopulation [  do-plots-population  ]
```

```
  if PlotAVGReputation [
     ifelse RepPlotMethod = "AVERAGE"
        [ do-plots-avgreputation  ]
        [ do-plots-samplereputation ]
  ]
  if PlotFitness [
     ifelse FitPlotMethod = "AVERAGE"
        [ do-plots-avgfitness ]
        [ do-plots-samplefitness ]
  ]
  do-plots-encounterhit
  do-plots-errorrate
  display
  ;; do-measure-all
end
;; ------------- END LOOP --------------------


to do-measure-all
  if ( ( count turtles with [ behaviourid = 1 and diestatus = False ])
< minturtle ) and diehonest = 400
    [ set diehonest tick ]
  if ( ( count turtles with [ behaviourid = 2 and diestatus = False ])
< minturtle ) and diehonest = 400
    [ set diefair tick ]
  if ( ( count turtles with [ behaviourid = 3 and diestatus = False ])
< minturtle ) and diehonest = 400
    [ set dieunpredictable tick ]
  if ( ( count turtles with [ behaviourid = 4 and diestatus = False ])
< minturtle ) and diehonest = 400
    [ set diedishonest tick ]
end

to do-plots-errorrate
  let sellers count turtles with [ asSeller = True ]
  let FPositive 0
  let FNegative 0
  if sellers > 0 [
      set FPositive (( count turtles with [ FalsePositive = True ]  / sellers) * 100)
      set FNegative (( count turtles with [ FalseNegative = True ]  / sellers) * 100)
  ]

  set-current-plot "Error Rate"
  set-current-plot-pen "FPositive"
  plot FPositive
  set-current-plot-pen "FNegative"
  plot FNegative

end

to do-plots-encounterhit
   let encounters ( count turtles with [ encountered = True ] /
count turtles with [ diestatus = False ]) * 100
   set-current-plot "EncounterHit"
   plot encounters
   set ecRate ecRate + encounters

end

;; ------------- START PLOT POPULATION GRAPH -----------
to do-plots-population
  set-current-plot "Population"
  set-current-plot-pen "Honest"
  plot count turtles with [ behaviourid = 1 and diestatus = false ]
  set-current-plot-pen "Fair"
  plot count turtles with [ behaviourid = 2 and diestatus = false ]
  set-current-plot-pen "Unpredictable"
  plot count turtles with [ behaviourid = 3 and diestatus = false ]
```

```
  set-current-plot-pen "Dishonest"
  plot count turtles with [ behaviourid = 4 and diestatus = false ]
end
;;

to do-plots-samplereputation
  set-current-plot "Reputation"
  set-current-plot-pen "Honest"
  plot mean opinions-of ex1
  set-current-plot-pen "Fair"
  plot mean opinions-of ex2
  set-current-plot-pen "Unpredictable"
  plot mean opinions-of ex3
  set-current-plot-pen "Dishonest"
  plot mean opinions-of ex4
end

to do-plots-avgreputation
  set-current-plot "Reputation"
  set-current-plot-pen "Honest"
  plot mean values-from turtles with [ behaviourid = 1  ] [ mean opinions ]
  set-current-plot-pen "Fair"
  plot mean values-from turtles with [ behaviourid = 2  ] [ mean opinions ]
  set-current-plot-pen "Unpredictable"
  plot mean values-from turtles with [ behaviourid = 3 ] [ mean opinions ]
  set-current-plot-pen "Dishonest"
  plot mean values-from turtles with [ behaviourid = 4  ] [ mean opinions ]
end



to do-plots-samplefitness
  set-current-plot "Fitness"
  set-current-plot-pen "Honest"
  plot fitness-of ex1
  set-current-plot-pen "Fair"
  plot fitness-of ex2
  set-current-plot-pen "Unpredictable"
  plot fitness-of ex3
  set-current-plot-pen "Dishonest"
  plot fitness-of ex4
end

to do-plots-avgfitness
  set-current-plot "Fitness"
  set-current-plot-pen "Honest"
  plot mean values-from turtles with [ behaviourid = 1  ] [fitness]
  set-current-plot-pen "Fair"
  plot mean values-from turtles with [ behaviourid = 2  ] [fitness]
  set-current-plot-pen "Unpredictable"
  plot mean values-from turtles with [ behaviourid = 3 ] [fitness]
  set-current-plot-pen "Dishonest"
  plot mean values-from turtles with [ behaviourid = 4  ] [fitness]
end




;; ------------ START MOVE -------------------
;; to make turtle moves randomly
to moverandomly
  fd random 4
  rt random-float 40
  lt random-float 40
end
;; ------------ END MOVE ----------------------
```

```
;; ------------- START TRANSACTING  --------------------
;; to make turtle transacts with closest neighbour
to tranVote
  set seller-turtle one-of other-turtles-here ; get the nearest turtle in the area
  if (seller-turtle != nobody and
      diestatus = false and diestatus-of seller-turtle = false ) [
    set encountered True
    set encountered-of seller-turtle ( True )
    set fitness-of seller-turtle ( fitness-of seller-turtle - EnergyUsed )
    set asSeller-of seller-turtle ( True )
    let LG 0
    ifelse RandomLG = true
      [ set LG  random-float(8) ]
      [ set LG LGRatio ]
    let Vote 0
    let P 0
    let PLeft 0
    let PRight 0
    foreach n-values totalagents [ ? ] [
      let pos ?
      if pos != theID [
        set P item pos opinions-of seller-turtle
        set PLeft ( P )
        set PRight ( ( 1 - P ) * LG  )
        ifelse PLeft > PRight
          [ set Vote (Vote + 1)
            if PlotDecision = True [
               set-current-plot "TrustDecision"
               set-current-plot-pen "Trust"
               plotxy LG P
             ]
          ]
          [
            if PlotDecision = True [
              set-current-plot "TrustDecision"
              set-current-plot-pen "NoTrust"
              plotxy LG P
              ]
          ]
      ]
    ]
    ifelse  Vote >= ((VoteThreshold + 1) / 100 * TotalAgents )
      [
         ifelse DecMethod = "VOTE+BROADCAST"
           [ goTransactingBroadcast ]
           [ goTransacting ]
      ][
         let l length(rangeoutcome-of seller-turtle)
         let posl random(l)
         let payoff item posl rangeoutcome-of seller-turtle
         if payoff >= GoodOutcomeThres [
             set falsePositive-of seller-turtle (True)
         ]
      ]
  ]
end
;; ------------- END TRANSACTING  ---------------------


to tranRecommendation
  set seller-turtle one-of other-turtles-here ; get the nearest turtle in the area
  if (seller-turtle != nobody and
      diestatus = false and diestatus-of seller-turtle = false ) [
    set fitness-of seller-turtle ( fitness-of seller-turtle - EnergyUsed )
    set encountered True
    set encountered-of seller-turtle ( True )
    set asSeller-of seller-turtle ( True )
```

```
    let LG 0
    ifelse RandomLG = true
      [ set LG  random-float(8) ]
      [ set LG LGRatio ]
    let P 0
    ifelse RepMethod = "BETA" [
          let g sum goodopinions-of seller-turtle
          let b sum badopinions-of seller-turtle
          set P ( ( g + 1 ) / ( g + b + 2 ) )
    ][
          set P ( sum opinions-of seller-turtle ) / ( totalagents - 1 )
    ]
    let PLeft ( P )
    let PRight ( ( 1 - P ) * LG  )
    ifelse PLeft > PRight
          [
            ifelse DecMethod = "VOTE+BROADCAST"
               [ goTransactingBroadcast ]
               [ goTransacting ]
            if PlotDecision = True [
               set-current-plot "TrustDecision"
               set-current-plot-pen "Trust"
               plotxy LG P
             ]
          ][
            let l length(rangeoutcome-of seller-turtle)
            let posl random(l)
            let payoff item posl rangeoutcome-of seller-turtle
            if payoff >= GoodOutcomeThres [
               set falsePositive-of seller-turtle (True)
            ]
            if PlotDecision = True [
              set-current-plot "TrustDecision"
              set-current-plot-pen "NoTrust"
              plotxy LG P
            ]
          ]
  ]
end


;; -------------- IF VOTED OK THEN TRANSACTING ---------
to goTransacting
  set fitness-of seller-turtle ( fitness-of seller-turtle + IncentiveFactor )
  let l length(rangeoutcome-of seller-turtle)
  let posl random(l)
  let payoff item posl rangeoutcome-of seller-turtle
  if payoff < GoodOutcomeThres [
     set falseNegative-of seller-turtle (True)
  ]
  ifelse RepMethod = "BETA" [
      let currentgood item theID goodopinions-of seller-turtle
      let currentbad item theID badopinions-of seller-turtle
      ifelse payoff >= GoodOutcomeThres
         [ set currentgood currentgood + IncentiveForGood ]
         [ set currentbad currentbad + SanctionForBad ]
      let P ( ( currentgood + 1 ) / ( currentgood + currentbad + 2 ) )
      set goodopinions-of seller-turtle
  replace-item theID goodopinions-of seller-turtle currentgood
      set badopinions-of seller-turtle
  replace-item theID badopinions-of seller-turtle currentbad
      set opinions-of seller-turtle ( replace-item theID opinions-of seller-turtle P )
   ][
      let currentopinion item theID opinions-of seller-turtle
      let newopinion 0
      ifelse payoff > GoodOutcomeThres
       [ set newopinion ( ( currentopinion + (payoff / 100)) / 2 ) ]
       [ set newopinion ( ( currentopinion + (payoff / 100)) / 2 ) ]
```

```
      set opinions-of seller-turtle replace-item theID opinions-of seller-turtle newopinion
   ]

end
;; ------------- END goTransacting ---------------------

to goTransactingBroadcast
  set fitness-of seller-turtle ( fitness-of seller-turtle + IncentiveFactor )
  let l length(rangeoutcome-of seller-turtle)
  let posl random(l)
  let payoff item posl rangeoutcome-of seller-turtle
  if payoff < GoodOutcomeThres [
     set falseNegative-of seller-turtle (True)
  ]
  foreach n-values totalagents [ ? ]
    [ if ? != theID-of seller-turtle [
         ifelse RepMethod = "BETA" [
            let currentgood item ? goodopinions-of seller-turtle
            let currentbad item ? badopinions-of seller-turtle
            ifelse payoff >= GoodOutcomeThres
               [ set currentgood currentgood + IncentiveForGood ]
               [ set currentbad currentbad + SanctionForBad ]
            let P ( ( currentgood + 1 ) / ( currentgood + currentbad + 2 ) )
            set goodopinions-of seller-turtle
replace-item ? goodopinions-of seller-turtle currentgood
            set badopinions-of seller-turtle
replace-item ? badopinions-of seller-turtle currentbad
            set opinions-of seller-turtle ( replace-item ? opinions-of seller-turtle P )
         ][
            let currentopinion item ? opinions-of seller-turtle
            let newopinion ( ( currentopinion + (payoff / 100)) / 2 )
            set opinions-of seller-turtle replace-item ? opinions-of
seller-turtle newopinion
         ]
     ]
    ]
end
```