High Quality P2P Service Provisioning via Decentralized Trust Management

THÈSE N^{O} 4711 (2010)

PRÉSENTÉE LE 31 MAI 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE SYSTÈMES D'INFORMATION RÉPARTIS
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Hung Le VU

acceptée sur proposition du jury:

Prof. A. Schiper, président du jury
Prof. K. Aberer, directeur de thèse
Prof. S. Buchegger, rapporteur
Prof. B. Faltings, rapporteur
Prof. L. Liu, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2010

This thesis is dedicated to my loving mother, the first person who has inspired my love and trust in the beauty of knowledge...

Acknowledgements

First and foremost, I would like to express my deepest thank to Prof. Karl Aberer, my thesis advisor, for his guidance and his total support for me since the first day I joined the lab. Karl is an excellent supervisor whose insight and vision has deeply inspired me in my research. I have learnt a lot from him during my PhD study and I am happy for having done my research in LSIR under his supervision.

I thank my thesis jury members, Prof. André Schiper, Prof. Boi Faltings, Prof. Ling Liu, and Prof. Sonja Buchegger, for their comments on my thesis, which have helped me a lot to improve the dissertation.

Prof. Manfred Hauswirth was the first person to welcome me to LSIR and co-supervised me during my first two years at EPFL. I am always grateful for what Manfred has been doing for me, both for of his mentoring my early research and for his supporting me in various practical aspects of my early life in Lausanne. Like Karl, Manfred is a wonderful mentor who I learnt a lot from, and still wish to have learnt more from him.

I also would like to thank Dr. Fabio Porto, a very good friend and a great colleague of mine. I graduated a lot from my work with Fabio: from him I learnt the way to overcome many research challenges and the way to really enjoy the life of an academics.

My deeply thank to Mrs. Chantal François, the secretary of LSIR, for her great work and supports, much more beyond her helping me with various logistics issues related to my conference travels and project meetings.

There are many other people I have worked with in various papers and research projects. In particular, I would like to send my thank and highest appreciation to Dr Zoran Despotovic, Dr. Thanasis G. Papaioannou, Prof. Sonja Buchegger, Prof. Anwitaman Datta, Wojciech Galuba, Doris Schiöberg, Othman Tajmouati, and Dr.

Sebastian Gerlach. Many pieces of my work in this thesis have been inspired and influenced from several interesting discussions and close collaboration with them.

It is so enjoyable to work in LSIR, and a big part of it comes from my wonderful labmates. Among them are Dr. Zoltan Miklos (my office mate), Dr. Oana Jurca, Dr. Gleb Skobeltsyn, Prof. Sebastian Michel, Dr. Adriana Budura, Dr. Roman Schmidt, Dr. Philippe Cudré-Mauroux, Prof. Yongluan Zhou, Dr. Hoyoung Jeung, and soon-to-be-doctors Surender Reddy Yerva and Nicolas Bonvin. Together we have had many enjoyable and interesting experiences that I will never forget.

I offer my truly thank to Marc-André Lüthi, Yves Lopes, and other members of the ICSIL1 computer administration group. They have done a great work in keeping the computing infrastructure of LSIR up and running smoothly, which made my research life significantly easier.

My life and work during the last five years would have never been complete without the helps and supports from my Vietnamese friends. I would like to give my sincere thanks to all of them, especially Dr. Nguyen Tuan Anh, Dr. Le Lam Son, To Lan Phuong, Dr. Luu Vinh Toan, Dinh Linh Chi, Dr. Nguyen Quang Huy, Dr. Vu Xuan Ha, Dr. Pham Van Thai, Dr. Vo Duc Duy, Ho Quoc Bang, Dr. Mai Tuan Anh, Dr. Nguyen Tuan Viet, Pham Minh Hai, Nguyen Minh Ha, To Huy Cuong, Bui Thi Kim Chung, Phan Van Anh, Do Lenh Hung Son, Dang Thai Ha, Le Ngoc Anh Thu, Dr. Nguyen Ngoc Vinh, Dao Tuyet Thao, Nguyen Hoang Minh, Nguyen Anh Thu, Ho Tran Long Dien, Dang Nguyet Minh, Hoang Ngoc Giang, and many, many more. Your helps and love, your supports and encouragements, and those wonderful moments I have had with you are beautiful memories that have always been a part of my life.

Lastly, I would like to express the most gratitude and thanks to my loving mother, for her silent but everlasting love for me, for her encouragement and supports, with which the writing of this thesis, an important chapter of my life, has been so much easier to complete.

Abstract

Trust management is essential to fostering cooperation and high quality service provisioning in several peer-to-peer (P2P) applications. Among those applications are customer-to-customer (C2C) trading sites and markets of services implemented on top of centralized infrastructures, P2P systems, or online social networks.

Under these application contexts, existing work does not adequately address the heterogeneity of the problem settings in practice. This heterogeneity includes the different approaches employed by the participants to evaluate trustworthiness of their partners, the diversity in contextual factors that influence service provisioning quality, as well as the variety of possible behavioral patterns of the participants. This thesis presents the design and usage of appropriate computational trust models to enforce cooperation and ensure high quality P2P service provisioning, considering the above heterogeneity issues.

In this thesis, first I will propose a graphical probabilistic framework for peers to model and evaluate trustworthiness of the others in a highly heterogeneous setting. The framework targets many important issues in trust research literature: the multi-dimensionality of trust, the reliability of different rating sources, and the personalized modeling and computation of trust in a participant based on the quality of services it provides.

Next, an analysis on the effective usage of computational trust models in environments where participants exhibit various behaviors, e.g., honest, rational, and malicious, will be presented. I provide theoretical results showing the conditions under which cooperation emerges when using trust learning models with a given detecting accuracy and how cooperation can still be sustained while reducing the cost and accuracy of those models.

As a another contribution, I also design and implement a general prototyping and simulation framework for reputation-based trust systems. The developed simulator can be used for many purposes, such as to discover new trust-related phenomena or to evaluate performance of a trust learning algorithm in complex settings.

Two potential applications of computational trust models are then discussed: (1) the selection and ranking of (Web) services based on quality ratings from reputable users, and (2) the use of a trust model to choose reliable delegates in a key recovery scenario in a distributed online social network.

Finally, I will identify a number of various issues in building next-generation, open reputation-based trust management systems as well as propose several future research directions starting from the work in this thesis.

Keywords: trust; reputation; quality of service; computational trust model; trust learning; peer-to-peer; service selection; service ranking;

Résumé

La gestion de la confiance est essentielle pour encourager la coopération et une haute qualité de service entre plusieurs applications pair-à-pair (P2P). Parmi ces applications, on dénote les sites commerciaux client-à-client (C2C), les marchés de services mis en œuvre à l'aide d'infrastructure centralisée, les systèmes de P2P, ou encore les réseaux sociaux en ligne.

Au regard de ces contextes d'application, l'état de l'art ne répond pas de manière adéquate à l'hétérogénéité des paramètres du problème en pratique. Cette hétérogénéité comprend les différentes approches adoptées par les participants à évaluer la fiabilité de leurs partenaires, la diversité des facteurs contextuels qui influencent la qualité de services, ainsi que la variété de comportements possibles des participants. Cette thèse présente la conception et l'utilisation de modèles de gestion de la confiance afin de renforcer la coopération et de garantir la haute qualité de service P2P, tout en tenant compte de l'hétérogénéité des paramètres évoquée ci-dessus.

Dans cette thèse, je vais tout d'abord proposer un framework probabiliste graphique pour les pairs afin de modéliser et d'évaluer la fiabilité des autres dans un environnement hautement hétérogène. Le framework répond à de nombreuses questions importantes de la littérature en recherche de gestion de la confiance: la multi-dimensionnalité de la confiance, la fiabilité des sources, et la modélisation personnalisée ainsi que le calcul de la confiance dans un participant basé sur la qualité de services fourni par ce dernier.

Ensuite, je présente une analyse de l'utilisation efficace des modèles de confiance dans des environnements où les participants présentent différents comportements, par exemple, honnête, rationnel, et malicieux. Je fournis des résultats théoriques montrant dans quelles conditions émerge la coopération lorsqu'on utilise des modèles d'apprentissage avec une précision donnée et comment la coopération peut encore être soutenue tout en réduisant le coût et la précision de ces modèles.

Comme autre contribution, j'ai aussi conçu et réalisé un framework de prototypage et de simulation pour les systèmes de gestion de la confiance basés sur la réputation. Le simulateur mis au point peut être utilisé à plusieurs fins, comme par exemple pour permettre la simulation empirique destinée à découvrir de nouveaux phénomènes en rapport avec la confiance ou pour évaluer la performance d'un algorithme d'apprentissage dans des environnements complexes.

Deux applications potentielles des modèles de confiance sont ensuite discutés: (1) la sélection et le classement des (web) services basés sur les appréciations d'utilisateurs de bonne réputation, et (2) l'utilisation d'un modèle de confiance pour choisir les délégués fiables dans un scénario de récupération de clé dans un réseau en ligne distribués social.

Enfin, j'identifie un certain nombre de questions diverses liées à la construction de systèmes de prochaine génération, aux systèmes de gestion de la confiance basés sur la réputation. Finalement, je propose quelques orientations futures de recherche dans ce domaine.

Mots-clés: confiance; réputation, qualité de service, modèle de confiance, apprentissage, pair-à-pair, sélection de services; classement de services;

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation of the Thesis

Trust is essential for the human society. As an intelligent living species on earth, we make trusting decisions when doing almost everything in life, sometimes without being aware of it. Since childhood, we trust our parents in their loving and caring for us. As grown-ups, we trust our friends and colleagues in assisting and cooperating with us to resolve difficulties in our personal and professional life. We even need to trust strangers in various complex situations: we believe that the goods we buy are worth the amount we paid, we trust the motorists to follow traffic regulations when going to work every morning, and that the laws would protect us in our daily transactions and business. In short, trust is vital to any human communication and interaction. Without trust, the (human) society would cease to exist.

According to Niklas Luhmann, a prominent thinker in sociological systems theory, the reason trust becomes so essential is due to the complexity of the society itself, when human beings face so many risks and uncertainty in almost every aspect of their daily activities. And thus trust is called on to reduce the complexity of society, as a result of an evolutionary process (Marsh, 1994).

The technological advances by several civilizations have made the human society become the most developed one that is known today on earth. As a matter of fact, the human society nowadays is even more complex and contains more risks and uncertainty than ever. As a result, it is no longer appropriate that we completely trust each other without any precondition in every situation. When it comes to more serious business people usually have incentives not to cooperate and behave towards their own interests, possibly at the cost of others. In a variety of situations, this opportunistic thinking and the lack of trust lead to the case where each individual and the whole community are worse off than if they would cooperate, e.g., the prisoner dilemma (Axelrod & Hamilton, 1981). To resolve such social dilemmas, it is long known that certain formal mechanisms are required to protect individuals against possible defections (Mui, 2002). Such mechanisms can be implemented in a variety of ways. For example, centralized authorities such as governments can use their enforcement resources to penalize malicious behavior and individuals who deviate from the systems of protocols, regulations, and laws specified by the society. Alternatively, the mechanisms can be implemented by the community as a whole by rewarding those members who are cooperative and penalizing (isolating) those that alleviate from the designed protocols and behavioral norms. For example, information on the behavior of a certain individual can be shared and spread from one member to another, e.g., Word-of-Mouth, and opinions of the whole community towards a certain member are aggregated into the so-called reputation of the member. This reputation would then be used by the community as a means to identify and reward good members as well as to isolate those ones with bad behaviors. Such above mechanisms are already in place, although informally, from the earliest times and was shown to be effective, e.g., in enforcing the cooperation and build trust among ancient Mediterranean traders (Jurca, 2007).

The wide adoption of the Internet, and with it the proliferation of online applications, has brought numerous opportunities to build business applications across organizational, political, and geographical boundaries. Examples of popular applications are online auctions, online shopping, peer-to-peer content sharing, and online social networking platforms, to name just a few. Along with the emergence of these online applications several challenges occur, of which one fundamental issue is the reliance of these applications on trust and voluntary cooperation among participating users. As a matter of fact, participants in any online applications need appropriate means to evaluate the trustworthiness of their potential partners whenever such partnerships may result in risky transactions or transactions with uncertain outcomes. For example, we often need to decide whether it is safe to pay for an article advertised by an unknown seller on a trading site such as eBay, or whether it is secure to download and install a piece of software from a content-sharing network. Consequently, more and more trust-related decisions must be made by users than ever.

Hard security techniques based on cryptographical primitives such as digital signatures may guarantee the non-repudiation of a participating partner and integrity of transferred information. However, these techniques do not help to assure the reliability and competence of the partner in successfully carrying out a transaction as expected. Online trust management mechanisms, the main subject of study of this thesis, offer a promising alternative and complementary approach to such issues.

Trust evaluation and management mechanisms, particularly those using reputation and user opinions (feedback), ensure that transactional partners would cooperate with each other, as desired by the community, in two different ways. First, the assessment of individual's trustworthiness helps to reveal the capabilities of different participants and enables users to select cooperative and more reliable partners for their transactions. Second, given the availability of a trust evaluation mechanism, individuals working in their own interests would find it optimal to behave according to the community norms, which usually favor cooperation over defection[1]. In fact, trust management can be considered as soft security approach that provide an additional layer of protection to participating users by ensuring the success of transactions carried out between members of the community.

The management of trust and fostering cooperation in virtual environments, however, is particularly challenging. First, online communities are usually without any centralized authority to enforce correct behavior and punish malicious participants. Even in systems under control of a specific provider, legal enforcement across political and geographical borders is costly and thus virtually impossible. Furthermore, users in online communities are identified only by pseudonyms, which are usually easy to obtain and change when desired. Consequently, reputation-based trust management systems are vulnerable to several attacks (A. Baker et al, 2007). Among these vulnerabilities, the most representative ones are ballot-stuffing and badmouthing attacks, where numerous Sybil identities (Douceur, 2002) collusively vote for or against certain users, and whitewashing behaviors exploiting the issue of cheap identities (Friedman & Resnick, 2001).

Despite these challenges, feedback-based trust management systems have found their practical application in almost every online collaborative system, from product comparisons and recommender systems to Web page ranking and social search (see Chapter 2 for more details). A large number of commercial initiatives on using trust mechanisms in other settings have been launched, such as to evaluate credibility of blogs[2], preventing spams and viruses[3], or to evaluate

[1] From an economist's viewpoint, these roles can be attributed to the use of reputation information in a trust management mechanism to address the issue of adverse selection and moral hazards in a system of self-interest agents (Dellarocas, 2005b).

[2] http://www.newscred.com/

[3] http://www.ciphertrust.com/products/trustedsource

reputation of potential business partners[1].

Research on trust management in decentralized systems, however, lacks of theoretical analysis to understand a variety of phenomena. For example, the majority of existing reputation-based trust mechanisms work very well in practice irrespective of their simplicity, e.g., the eBay rating system. As a matter of fact, current work in the trust research community is narrowly focused on experiments enlightening only part of the overall picture, while general properties are not fully understood (Abdul-Rahman, 2005). The goal of this thesis is to provide a more complete understanding of such phenomena in heterogeneous scenarios with complex settings. Particularly, I want to understand the principles and nature of these (reputation-based) trust mechanisms as well as their possibilities and limitations, and to answer fundamental questions related to the design of trust mechanisms for different practical applications. For example, the extent to which a trust mechanism should be used, and which properties of a trust measure are necessary to ensure high cooperation among self-interested agents with various types of behaviors is one of the key issues addressed in the thesis.

1.2 Scope of Research

This research work focuses on ensuring high quality service provisioning in peer-to-peer (P2P) environments, where a participating user (a peer) plays different roles in the system: a service (or resource) provider, a service user, or a rater to evaluate quality of a service. An abstraction of a P2P application is given in Fig. 1.1.

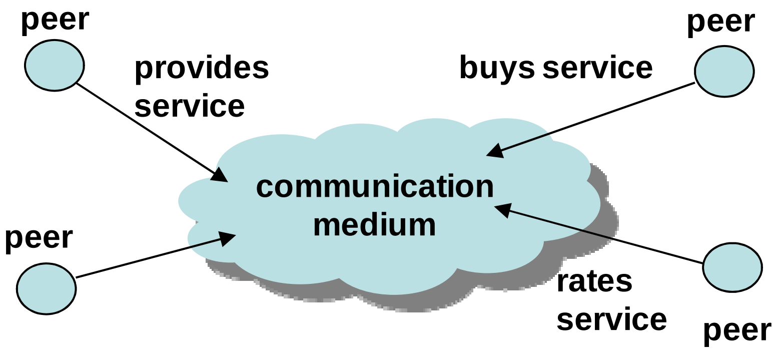


Figure 1.1: Service provisioning in peer-to-peer applications

[1] http://www.rapleaf.com/

A peer may provide a service at different prices and at different quality levels, e.g., either high or low quality. Only services with high quality are desired to a service user (another peer). Peers may leave or join dynamically, and in some cases there may exist identity management schemes to ensure that identities are not cheap. The communication medium to share information among peers may be centralized, e.g., the case of eBay, or decentralized, e.g., a storage system on top of a P2P network. Note that even if the communication medium is provided by a centralized entity, interactions among different participants are still peer-to-peer in nature.

In this thesis, we study the problem of building trust and fostering cooperation among the participants in a P2P application, in consideration of their possible behaviors in the application layer. The underlying support layers such as data storage and communication among different peers are assumed to function correctly. That is, we assume that P2P applications are built on top of a (decentralized or centralized) data management infrastructure that supports the storage and retrieval of information efficiently, reliably, and securely. Particularly, we do not consider the problem of peers maliciously manipulating requests or dropping packets when sending and receiving messages. Similarly, data stored in the system is assumed not to be tampered with and can be retrieved relatively easily with insignificant delay, e.g., via the use of (distributed) search methods and existing cryptographical protocols such as symmetric encryption and digital signatures. Although trust management mechanisms can also be used to address these types of misbehavior, solving cooperation and trust-related issues in every layer of the system is beyond the scope of this thesis.

The above abstraction represents different practical online business applications such as trading on eBay and on social networks, peer-to-peer content sharing, and markets of (Web) services. The inherent problem therein is that peers usually have strong incentives not to cooperate. That is, a peer as a service provider may provide low quality service, by not contributing resources (free rider) or not shipping an item after receiving payment (cheating/fraud in trading). In such cases simple reputation systems, e.g., by allowing peers rate partners after transaction, may not be sufficient, since users may be malicious and ratings are thus not always reliable. In more complex scenarios, certain users are also rational and only cooperate when necessary to maximize their benefits. Our goal is to understand how reputation builds trust and enforces cooperation between transactional partners in such an application framework with heterogeneous settings.

Since the seminal thesis of Marsh (Marsh, 1994), there have been numerous work dedicated to propose different trust evaluation mechanisms and their applications in a variety of contexts, resulting in many dissertation works over the recent years, namely (Abdul-Rahman, 2005; Despotovic, 2005; Jurca, 2007; Kohlas, 2007; Krukow, 2006; Levien, 2002; Mui, 2002; Sabater, 2003a; Teacy, 2006; Xiong, 2005). What differentiates this work from other research is that I do not focus on developing specific, ad-hoc trust models for a given application scenario. Instead, the work in this dissertation follows a so-called meta-approach:

- I survey several existing (reputation-based) trust management systems to identify the questions addressed by these approaches, their common working principles, and their underlying assumptions. Based on this extensive review I propose an overall probabilistic framework that can be seen as a generalization of many existing works on building computational trust models. Several approaches can be developed and analyzed under this umbrella framework.
- Existing gaps between work in the trust management literature are studied and solutions to complete these gaps are proposed. By studying the relation between accuracy of a computational trust model and its possibility to enforce trust, my work provides a starting point to use existing probabilistic computational trust models in systems where users exhibit both probabilistic and rational behaviors. By analyzing the feasibility of applying reputation-based trust mechanisms when cheap identities are possible, this thesis proposes ways to apply existing computational trust models in an even wider range of scenarios.
- I also study various requirements and suggest solutions for the development of next-generation, open reputation-based trust management systems based on possible extensions of contemporary systems: by opening them up and combine their capability for overall performance improvement. Different challenges and opportunities for such possible extensions are studied in detail as a starting point for future research.

1.3 Research Contributions

This thesis provides the following research contributions:

1. I have proposed a graphical probabilistic framework that helps the personalized modeling and learning of trust and quality for a given application based on domain knowledge. The framework is shown to have many advantages: it is generic for many scenarios, such as for modeling and learning quality of services with multi-dimensional quality properties, the

output has well-defined semantics and useful meaning for many purposes. The implementation has scalable performance, computational, and communication cost, and produces reasonably good estimates, even with a sparse and incomplete recommendation data set.

Besides, this framework gives us two interesting findings. First, it is shown that modeling is crucial in the design and implementation of a computational trust model, especially when trust is defined on multi-dimensional qualitative behavior of agents, and where ratings are collected from information sources with unknown reliability. Domain knowledge and human experts are very important for the design of a good generative model for trust learning. Second, the proposed framework generalizes many existing computational trust models: they are in fact developed based on different generative models of the user's behaviors, and trust evaluation is done by learning on these models using different heuristics. Therefore, given this umbrella framework many computational approaches to trust evaluation can be developed and readily analyzed using available machine learning tools.

2. I provided the design and implementation of a prototyping and simulation tool for trust evaluation algorithms, based on the abstraction obtained from the above graphical probabilistic framework. This simulation platform is useful for many purposes.

   First, it can serve as a flexible and extensible experimentation platform supporting empirical simulation to discover new phenomena or to test various hypotheses. For example, I have been using this simulation environment to measure the cooperation level in various scenarios where peers use different learning algorithms and with various behaviors, including honest, malicious and rationally strategic. Second, the simulator can be used as a competition test-bed, similar to the ART test-bed (Fullam *et al.*, 2005) in AI communities with much more flexible testing environments. E.g., different application scenarios can be developed and deployed to test performance of different algorithms.

   Third, the framework can be used as a prototyping tool: basic building blocks provided by the framework help one to develop and test performance of his or her trust learning and decision making algorithms under a variety of standard and/or customized testing scenarios with much less development effort. The implemented trust management algorithms can also be used as an application-independent library of reputation-based trust learning algorithms to be used in a specific scenario. The modular, lego-block design of the trust simulation framework enables different implementations and even makes emulation of real scenarios possible.

3. I performed a detailed analysis of possibilities of using a computational trust model in open and decentralized environments where participants exhibit different behaviors, including intrinsically honest, rationally strategic and malicious. I provide theoretical results that show under which conditions cooperation emerges when using trust learning models with given accuracy and how cooperation can be still sustained while reducing cost and accuracy of those models. I prove that such a model with reasonable false positives and false negatives can effectively build trust and cooperation in the system, considering rationality of participants. These results reveal two interesting observations.

   First, the key to the success of a reputation system in a rational environment is not a particular sophisticated trust learning mechanism but an effective identity management scheme to prevent whitewashing behaviors. Hence, in such environments it is more important for researchers and practitioners to focus on developing better identity management techniques rather than on complicated computational trust models to detect misbehavior.

   Second, given an appropriate identity management mechanism, a reputation-based trust model with moderate accuracy bound can be used to enforce cooperation effectively in systems with both rational and malicious participants. In heterogeneous environments where peers use different algorithms to detect misbehavior of potential partners, cooperation still emerges as long as these algorithms have a moderate misclassification error bound.

4. I proved that even if identities are cheap, there exists mechanisms to deter whitewashing behaviors with little inefficiency. By using an identity premium-based pricing mechanism, under certain application-specific assumptions, rational peers find it optimal to cooperate in providing every but their last service to the system.

5. I proposed two novel and practical applications of computational trust models. The first application is to rank services based on their quality properties. These quality ratings are collected from many users, and only reports of reputable users are considered in the evaluation of the service quality. This work, published in (Vu *et al.*, 2005a), is one of the earliest work on using reputation-based trust management in service-oriented environments. It is verified experimentally that if trust is placed correctly on a relatively small number of users, it is possible to eliminate a large number of malicious users colluding to manipulate the ranks of certain services.

The second application of computational trust models is the use of a dedicated trust models to evaluate and to select potential delegates in a key backup-recovery sharing scenario on top of a distributed social network. Therein the service provided by each peer is the secret keeping service, and the trustworthiness of a peer is well correlated to its reliability in keeping the share secretly from attempts to steal the secret from an adversary.

6. Possibilities of building next-generation trust management systems are also investigated. I have given a first systematic study on various challenges and opportunities of integrating and enabling exchanges of information among similar reputation-based trust systems to improve the overall system capabilities in detecting malicious behavior and enforcing cooperation. Many interesting research problems are identified. These include the fair and privacy-preserving exchange of information among systems, possible ways to integrate malicious behavior detection algorithms with evidences collected from many user communities, and how different levels of system openness affect the performance of the integrated reputation system.

1.4 Thesis Outline

This thesis is structured mostly for readers' convenience: although each chapter addresses certain issues that are well related to other chapters, each can be read independently.

This first chapter presents the motivations of this thesis, introduces the structure of the thesis, and gives a list of research contributions. Chapter 2 provides background information, and briefly surveys the research literature on trust management in open and decentralized systems. Important open research problems are then identified. Some of these open questions are the topics this thesis will address.

Chapter 3 studies the design and implementation of a reputation-based computational trust mechanism from a machine learning perspective. In this chapter first I present a graphical probabilistic framework that enables personalized modeling and learning of trust and quality in a given application based on available domain knowledge. Next, I present the design and implementation of tool for prototyping and simulation different computational trust evaluation algorithms. This prototyping and simulation framework can be used for many purposes: as a simulation testbed, a prototyping framework, or as a library of commonly used algorithms.

Chapter 4 presents an abstraction of different computational trust models and studies the relation between accuracy of a trust model and its capability to deter bad behaviors and promote cooperation. This relation is analyzed in the context of an (open and decentralized) peer-to-peer e-trading system with and without an effective identity management in place, where cheap pseudonyms may be possible.

Chapter 5 presents some applications of (reputation-based) computational trust models. The first application is on evaluating service quality and ranking services based on their quality properties. The second application uses a specialized trust measure as a means to select most reliable delegates to keep secrets of users in a distributed social networks.

Chapter 6 studies the building of next generation reputation-based trust systems, as part of my on-going work. I present the challenges and opportunities of building more robust trust systems via opening and integrating different reputation systems.

Chapter 7 concludes the thesis and provides pointers to future research in decentralized trust management. In the appendix are some examples of probabilistic inference algorithms, my resume, and the publications related to this thesis.

Chapter 2

Overview of Decentralized Trust Management in Open Systems

This chapter explains the basic concepts of a trust management system and provides an overview of trust-related research on open and decentralized systems over the recent years. A detailed review, however, is not given in this thesis, as there are already several such surveys in the literature (Despotovic & Aberer, 2006; Golbeck, 2006; Jøsang *et al.*, 2007; Wang & Lin, 2008). In fact, trust management has generated an incredible number of research works over the recent years across so many diversified areas and a thorough survey deserves a book of its own. Therefore, the goal of this chapter is to summarize the most common research themes and point out those research questions that are addressed in this thesis. For each of the forthcoming chapters the most relevant work will then be presented in the context of each chapter's topic.

2.1 Fundamentals of Trust Management Systems

The basic operational concepts of a trust management system are given in Fig. 2.1. Three key actors in any trust system are a *trustor*, a *trustee*, and a *computational trust model* to be used by the trustor.

A trustor is a participant who needs to evaluate its belief on whether a potential partner (a trustee) behaves according to its expectation. A computational trust model is a computational mechanism that operates on the evidences related to the trustworthiness of the trustee and outputs an appropriate trust measure. This trust measure is interpreted by the trustor as an indication whether to engage the trustee in a transaction (*trust establishment*). Trust manage-

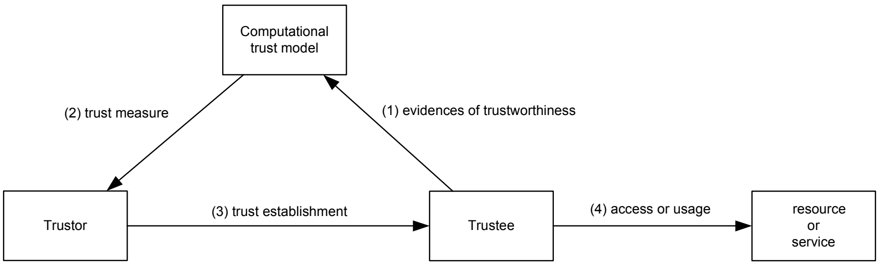


Figure 2.1: Trust management systems: basic concepts and operations.

ment is a necessity in those systems where the engagement of the trustee in such a transaction brings the trustee a positive gain at a potential loss of the trustor (Despotovic, 2005).

Any participant in the system may play the role of either a trustor or a trustee in different contexts. The nature of a transaction between a trustor and a trustee, e.g., by giving the trustee access to a resource or service, is application-dependent. The mentioned service or resource can be provided by either a trustor or a trustee in many forms. For example, in an e-trading application where the trustor is the buyer and the trustee is the seller of an item, the resource is the payment of the buyer to the seller before the item is shipped. In a company's computer network, the trustor is the system manager who evaluates whether to grant the an employee access to a meeting room or the right to use expensive devices for preparation of the meeting.

Such a trust management system employs several basic conepts, most notably a *trust measure semantics* and the implementation of the corresponding *computational trust models*. Several computational representations of trust have been developed: trust can be measured as a probability (Despotovic & Aberer, 2005), as a (standard or subjective) logical fact in a knowledge-base (Jøsang *et al.*, 2006), or as a scalar value to be used as a ranking metric (Kamvar *et al.*, 2003; Yu & Singh, 2000).

Trust management systems are using different types of *trustworthiness evidences*. The definition of such evidences determines how the computational trust model is built and which trust measures to be used in the system. Trustworthiness evidences can be credentials of the trustee as demanded by the trustor. For example, digital signatures and keys are main evidences in policy-based trust management systems, such as PolicyMaker/KeyNote (Blaze *et al.*, 1996). In reputation-based trust systems (Despotovic, 2005), these evidences are (quality) ratings by previous users on the service the trustee provides, and related information relevant to the evaluation of the trustee's trustworthiness. Examples of such information are the (social) relationship between the trustor and the trustee, or the popularity of the provided service or resource.

A trust management system can be either centralized or decentralized depending on the way the computational trust model is implemented. In centralized trust management systems, there exists a single computational trust model used by a central management entity to evaluate the trustworthiness of every participant in the system. This central management entity is implicitly trusted by every user in the system in its ability to collect and estimate the trustworthiness of every other user, e.g., in access-control systems (Chapin *et al.*, 2008). On the other hand, participants in a decentralized trust management system use a variety of computational trust models, possibly implemented with different trustworthiness evidences according to preferences of each participant to evaluate the trustworthiness of a potential partner.

In this thesis, we consider the problem of ensuring that services provided by a trustee to a trustor will be of high quality via the usage of appropriate trust management approaches. Such service provisioning is done in a P2P fashion, that is, a participant can be either a provider or a consumer of a service in different transactions. Trust is understood and interpreted in this thesis as the the probability that estimated by a trustor that a trustee behaves as the trustor expects. We consider a heterogeneous system, where different computational trust models can be implemented and used by different participants. Each computational trust model can either be a centralized entity or a local computational mechanism employed by the trustor to evaluate the trustee's trustworthiness. The main parts of this thesis focus on designing and implementing a computational trust model, study its main desired properties, the way to use it effectively in environments where participants (both the trustors and trustees) have different types of behavior, including honest, strategic, and malicious.

2.2 Decentralized Trust Management Approaches

A (somewhat coarse) taxonomy of existing decentralized trust management systems is given in Fig. 2.2. Depending on whether the trustor uses only the identification of the trustee as the major decisive factor in trust evaluation, trust systems are roughly classified into two main categories: *credential* or *policy-based* vs. *reputation-based* trust systems.

In a policy-based trust management systems, a trustor establishes its trust on a trustee if and only if the trustee has shown to possess certain credentials, e.g., keys, digital signatures, or certificates. Provided its credentials satisfy requirements of the system security policies, a participant may gain access to the defined resources. Tools and techniques used in the

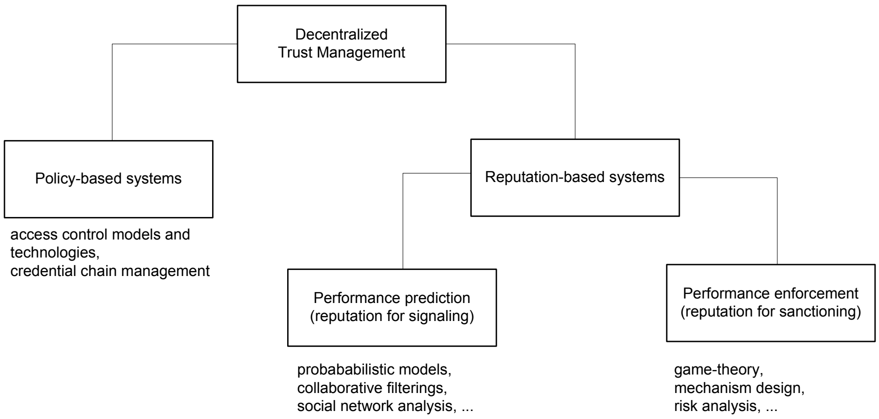


Figure 2.2: A taxonomy of decentralized trust management systems.

development of policy-based trust management systems are credential-chain management and access control models and technologies. Popular approaches for this type of trust management system are PolicyMaker and its descendant KeyNote (Blaze *et al.*, 1996).

Policy-based trust management systems are not the focus of this thesis and thus we will not review research issues related to this type of systems. For interested readers, a thorough survey of trust management systems based on policy and security measures can be found in (Chapin *et al.*, 2008).

A major disadvantage of policy-based trust management systems is their inflexible approach to estimate trust between participants. Trust between a participant to another is usually defined based on the claimed identity and credentials of the latter, rather than on its actual performance and reliability. Such a trust model limits applications of policy-based systems to those scenarios where the policy maker is trusted to always issues *perfect rules* to capture behaviors of every participant. This is not always true in dynamic environments where participants possessing certain credentials usually have incentives to deviate from the cooperation protocol to maximize their benefits. For example, using a policy-based trust system, it is possible and reasonably safe to state that in a hospital, someone proven to be a doctor has the right to access and use all medical equipments in the operation rooms. Such a policy is simple to formulate via deterministic rules given a description of the resources and actors in the system, as well as the (mostly static) relation among them. However, it would be more difficult to write similar policies to determine which groups of the available doctors at the present time have the best experience and expertise to supervise or carry out a sophisticated surgery on a specially important person.

The second scenario mandates that the system must use a more sophisticated mechanism to evaluate the trustworthiness on those doctors available. In such cases the decision to trust a doctor should be based on much more information, namely on their experience, their actual performance, and even their (professional) relationship with other doctors in the surgery team. More expressive languages (normally logic-based) can be used to develop more complex security and trust policies, yet the possibly high complexity of their evaluation is the main challenge in many existing systems (Chapin *et al.*, 2008).

Reputation-based trust systems offer a more complete alternative solution by enabling a trustor to evaluate and place trust on different other trustees according to various performance-oriented metrics based on historical data on the behavior of these trustees.

Note that in practice, *hybrids* of reputation and policy-based trust systems are possible: a production system may employ both policies and reputation information for trust evaluation. For example, historical performance statistics of different entities can be collected and stored in a system-wide (and distributed) database. With the help of expressive high-level query and (normally logic-based) policy languages, entities can be evaluated in terms of reliability and then be authorized to do critical tasks, e.g., as in QCM and its successor SD3 (Jim, 2001). Similarly, computational trust models used in a reputation-based system can be implemented to incorporate those policies determined by a policy-based trust system to ensure that the security policies of the system are also complied with.

Reputation-based trust management has attracted lots of research efforts over the recent years across several research domains, from sociology to (micro)-economics, and computer science. A large body of literature has developed around the notion of using computational trust models in open and decentralized systems (Dellarocas, 2005b; Despotovic & Aberer, 2006; Ding *et al.*, 2009; Golbeck, 2006; Jøsang *et al.*, 2007; Wang & Lin, 2008). Roughly speaking, there are two solution classes for reputation-based trust management systems, based on different viewpoints (Dellarocas, 2005b; Despotovic & Aberer, 2006) (see Fig. 2.2).

- **Performance prediction approach**: systems using this solution class assume probabilistic behavior of the participants. They focus on development of computational models that are capable of learning these behaviors to predict performance of individuals or groups of participants. Various machine learning tools and heuristics such as probabilistic networks, collaborative filtering, social network analysis, are employed to predict future performance of participants and to select the best potential partners for forthcoming transactions. For example, peers behaving as honest over time are considered to be honest in future transactions with certain positive probabilities. Detecting malicious behaviors is also one of the main targets of works in this class. Therefore, reputation information is used in those systems as a means to *signal* the community on the performance of a participant. Representative seminal work in this category includes (Aberer & Despotovic, 2001; Kamvar *et al.*, 2003; Xiong & Liu, 2004).

- **Performance enforcement approach**: This solution category comprises those approaches that address the rational behaviors of the participants. The main assumption herein is that participants behave to maximize their expected utilities, and following game-theoretical reasoning rules. Thus, research works in this class aims at designing games or interaction protocols where full cooperation is the only (Nash/Pareto-optimal) equilibrium for every participant. Game theory, mechanism design, and risk analysis methods are usually used in works of this category. Reputation information is mainly used as a tool to *sanction* bad behavior and to promote cooperation. Examples of representative work following this direction are (Dellarocas, 2004, 2005a; Jurca, 2007; Miller *et al.*, 2005).

The majority of work on trust management over the recent years focuses on developing dedicated computational trust models for a variety of application scenarios. These works are tested and evaluated via several ad-hoc experiment tools and simulators and in most cases, no major theoretical foundation and analysis have been developed. We will not focus on reviewing these works in this chapter, as such an attempt has been made in several existing surveys (Dellarocas, 2005b; Despotovic & Aberer, 2006; Ding *et al.*, 2009; Golbeck, 2006; Jøsang *et al.*, 2007; Wang & Lin, 2008). Instead, we are going to focus on the most major and new issues that have been addressed by the trust-related research over the recent years. Base on these observations, we will identify the most important and recent issues to be addressed in this thesis. A (non-exhaustive) summary of research issues targeted over the recent years, since the start of this thesis, includes work on:

- **Incentive-compatible trust management mechanisms**: An interesting approach to reputation-based trust management is to discourage non-truthful reporting using game-theoretic principles. Milller, Resnick and Zeckhauser (Miller *et al.*, 2005) propose to reward reputation reports as a function of their estimated truthfulness, and show that this makes truthful reporting an equilibrum strategy. These results have been significantly strengthened by Jurca and Faltings to minimize the payments and make truthful reporting the highest-paying Nash equilibrum (Jurca, 2007). However, collecting real data for empirical experiments on the effectiveness of these approaches on real human users, e.g., their rational choice and preferences, is an interesting problem not fully investigated. For example, (Schosser *et al.*, 2008) performs some controlled experiments on groups of people and concludes that humans tend to find it difficult to resort to the strategies expected by the system designer.

- **Model-based trust learning:** Several computational trust models have been developed using a variety of machine learning techniques. The problem of trust modeling is identified to be more important, especially when trust is a multi-dimensional concept and domain knowledge should be incorporated to improve the learning (Vu & Aberer, 2007, 2009). Along this research line, various works propose the learning and evaluation of trust based on building appropriate generative models of user behaviors (Hsu *et al.*, 2006a; Regan *et al.*, 2006; Wang *et al.*, 2006). (Rettinger *et al.*, 2008) models interaction among agents according to a Hidden Markov model and predicts best actions of agents according to the model, which is trained from the system transaction logs.

- **Reputation trust models for groups**: In many naturally formed robust community-based networks, for example, "swarms" in P2P file sharing systems, organizational structures of many online environments (forums, e-markets, etc.), one can easily observe a much higher level of trust and cooperation within a community than with the outside. An interesting approach is proposed in (Gupta & Somani, 2004), where the authors view reputation of a group or individual as currency to be exchanged in the network. Other work defines groups of users based on various heuristics, such as based on their common interests and transactions (Ashri *et al.*, 2005).

- **Comparing performance of different computational trust models:** Since most existing computational trust and reputation models use similar principles, they can be compared to determine the most effective mechanism under different attack models. Towards this direction we mention the development of the ART testbed (Fullam *et al.*, 2005) as a competition venue for comparing many trust evaluation algorithms. (Bryce *et al.*, 2005; Kinateder *et al.*, 2005) define general trust metrics to quantify the effectiveness of the results as a basis for determining the theoretical/practical limitation of a reputation mechanism. (Suryanarayana *et al.*, 2006) proposes a software architecture framework to enable the easy realization and integration of a reputation system into a new application. (Liang & Shi, 2008; Schlosser *et al.*, 2005) analyze the effectiveness of different reputation mechanisms via empirical simulations. (von der Weth & Böhm, 2007) compares different centrality measures, i.e., different computational models, to rank individuals based on feedback and concludes that no single measure excels in most cases. Modeling and generating malicious attacks for empirical simulations and testing the robustness of a trust systems are also studied recently in (A. Baker et al, 2007; Yang *et al.*, 2009).

- **Sybilproof and cheap identities in reputation systems**: (Landa *et al.*, 2009) studies the ability of a reputation system to counter Sybil attacks (Douceur, 2002) and provides some negative results. This means identity management still remains as the major open issue in designing and implementation of future reputation-based systems.

Beside, a large number of research works over the recent years have focused on finding practical applications of computational trust models, among which the most representatives are:

- **Quality-based ranking and selection of services and resources**: reputation-based trust models have been used both in service-oriented architectures to benefit for the selection and composition of services based on their quality properties (Kalepu *et al.*, 2004; Maximilien & Singh, 2002, 2005; Vu *et al.*, 2005a). A similar application is in Grid resource discovery and selection (von Laszewski *et al.*, 2005).
- **Quality-based information retrieval**: these works include several attempts to use user's trust management to enhance the correctness of business recommender systems, namely (Avesani *et al.*, 2005; Jamali & Ester, 2009; Massa & Avesani, 2004), to discover most reliable information Web sources, such as (Yin *et al.*, 2007). Search engines such as Google have also integrated feedback into their search results by enabling users to rate and modify ranking results.
- **Eliminating malicious behaviors** in various applications, such as filtering of mail and web spamming (Golbeck & Hendler, 2004; Gyongyi *et al.*, 2004), and in estimation of the reliability of Wikipedia entries (Adler & de Alfaro, 2007). Other types of softwares, such as antivirus programs, have begun to use reputation information to identify most vulnerable files and system components in their recent releases.
- **Enforcing cooperation in open protocols**: Reputation and trust management has been shown to be effective in enforcing cooperation in many system where participants find it easy and have strong incentives to deviate from the designed protocols. These works include the following (non-exhaustive) application areas: routing in mobile/ad-hoc/sensor networks (Buchegger & Boudec, 2004; Galuba *et al.*, 2007; Twigg, 2003),

sensor position estimation (Kiyavash & Koushanfar, 2007), WiFi deployment (Salem *et al.*, 2005), network selection and ranking (Shubha Kher & Somani, 2005), avoiding free riders in P2P file sharing, Web caching and Web sharing systems, human-network based filtering system (Liang & Shi, 2005), Grid task scheduling (Zhao *et al.*, 2005), etc.

2.3 Research Issues Addressed in the Thesis

Heterogeneity in computational trust models employed by participants

The first open issue that is addressed in this thesis is the heterogeneity in the approaches employed by the participants to evaluate trustworthiness of their partners. This heterogeneity origins from many reasons. Peers may have different personalized preferences, different available information and prior beliefs on the trustworthiness of the rating users. The decentralized nature of a system implies that peers may develop a variety of personalized computational trust models based on the available information.

Additionally, in a heterogeneous environment, peers can also offer various types of services to the others, each with different quality attributes. Service provisioning quality is influenced by a diversity of contextual factors. In this case trust in a peer is mostly based on the quality it offers to the others, and thus is a multi-dimensional concept. Correctly modeling and predicting trustworthiness of peers (in terms of service quality) is non-trivial, as observations and ratings on the service quality on a peer could only be collected from many different observers with different reliability and possibly with many missing values. It is our interest to study the existing approaches and generalize them into a framework that allows the easy development and analysis of different learning algorithms to model and evaluate quality of services provided by different peers. This problem is presented and studied in detail in Chapter 3 of the thesis. Given the availability of this framework, depending on the available information, computational resources, and preferences, each peer in a heterogeneous system can easily apply a suitable learning algorithm to accurately model and predict service quality of others whenever such an evaluation is necessary to select the best partner for future transactions.

Heterogeneity in possible behavioral patterns

Besides the diversity in trust evaluation approaches employed by peers in the system, participating users may also have different behavioral patterns. The majority of work on trust management considers only malicious or rational participants in a rather homogeneous system.

The diversity in the (bounded) rationality and maliciousness of peers in the system, is still not fully explored. Therefore, it is also important to study whether under such heterogeneous settings whether cooperation also emerges. The second question this thesis addresses is the effective usage of computational trust models in such an environment. Specifically, we consider the case where participants exhibit different behavioral patterns, including intrinsically honest, opportunistically rational, and malicious (Chapter 4).

Building a general simulation framework for trust-related research

Another issue in decentralized trust management is the simulation and evaluation of various computational trust models. The main reason is that most existing works on trust research rely on simulation to verify their performance and effectiveness. Even if a data set from a real reputation and recommender system can be used, usually additional synthetic data need to be generated, e.g., to simulate various attacks and to discover possible vulnerabilities of the system. Such experiments are usually conducted in different ad-hoc simulation environments, making comparison of existing solutions extremely difficult. As another contribution, I designed and implemented a general prototyping and simulation framework for reputation-based trust systems. The developed simulator can be used for many purposes, such as to enable empirical simulation to discover new phenomena or to test various hypotheses in complex settings. Additionally, the basic building blocks provided by the framework may help researchers to test the performance of their trust learning approaches, under a variety of customized testing scenarios, with minimal development effort. This work is a first step towards building a test benchmark for the trust-related research communities (Chapter 3, Section 3.2).

Developing open trust management systems

Current reputation-based trust management systems that are used across different applications, e.g., e-trading systems, recommender systems, social networks, are closed in the sense that they do not share any information on the user communities, the computational trust models being used, etc. among each other. This is in stark contrast with the openness of the Internet architecture and its ecosystems. In fact, by sharing information among reputation-based trust systems with other online communities, users' digital footprints can be detected. This helps in the development of more effective identity management mechanisms that is less affected by the issue of cheap pseudonyms, i.e., prevents whitewashing behaviors. In this thesis, I also present a study of various research issues for building next-generation, open reputation-based trust systems. I study the sharing of information and combination of computational trust models from different reputation systems to achieve better performance in detecting misbehavior. Furthermore, I explore several possibilities and challenges in the development of such open trust management systems, and provide initial solutions to some of these problems identified (Chapter 6).

Chapter 3

A General Framework for Decentralized Trust Management

In this chapter, I propose a probabilistic framework targeting three important issues in the computation of quality and trust in decentralized systems. This approach addresses the multi-dimensionality of quality and trust, taking into account credibility of the collected data sources for more reliable estimates, while also enabling the personalization of the trust computation. I show that domain knowledge on structure of service provided by peers and related constraints, such as causal dependencies among quality attributes and contextual factors, while widely available, can be exploited to effectively address the above issues in a theoretically-sound framework. Specifically, I use graphical models to represent peers' qualitative behaviors and exploit appropriate probabilistic learning and inference algorithms to evaluate their quality and trustworthiness based on related reports. My implementation of the framework uses the Expectation-Maximization algorithm to learn parameters of the model and the Junction Tree Algorithm to infer from them the estimation of quality and trust. The framework highlights the importance of modeling in trust management, as several existing computational trust approaches can be shown to apply certain heuristic parameter learning algorithms on simple generative models identified in the framework.

The experimental results validate the advantages of my approach: first, using an appropriate personalized quality model, my computational framework can produce good estimates, even with a sparse and incomplete rating data set; second, the output of this solution has well-defined semantics and useful meanings for many purposes; third, the framework is scalable in terms of performance, computation, and communication cost. Furthermore, my solution can be understood as a generalization or serve as the theoretical basis of many existing trust computational approaches.

The second part of this chapter presents the design and implementation of a generic trust prototyping and simulation engine, named ARTS. This simulator enables the fast development of different trust learning algorithms based on the concepts identified by the above probabilistic framework. The current implementation of the simulator can be used to comparatively test performance of different reputation-based trust evaluation algorithms, and in fact it is used as the foundation of the experiments in the later chapters. The modular design of the simulator makes it possible to use different implementations of the simulation engine, e.g, a distributed simulation engine and even enable emulation of real application scenarios.

The work in this chapter is published at (Vu & Aberer, 2006, 2007, 2009; Vu *et al.*, 2005a).

3.1 A Graphical Probabilistic Approach to Decentralized Trust Modeling and Learning

Quality and trust are inter-related concepts that have become increasingly important factors in both our social life and online commerce environments. In many e-business scenarios where competitive providers offer various functionally equivalent services, quality is often the most decisive criterion that helps to build trust among participants in the system and influences their selection of most prospective partners. For example, between two file hosting service providers, a user would aim for the one allowing the storage of larger files in longer periods, offering higher download and upload speed under better pricing conditions. Similarly, there are several other types of services that are highly differentiated by their quality of service (QoS) features such as Internet TV/radio stations, online music stores or teleconferencing services. Therefore, appropriate mechanisms for accurate evaluation of service quality have become highly necessary.

In large scale decentralized systems with no central trusted authorities, past performance statistics are usually considered as indicative of future quality behaviors (Resnick *et al.*, 2000). An autonomous agent (or peer) acting on behalf of its users often uses ratings or recommendations from other peers to estimate QoS and trustworthiness of its potential partners before involving into costly transactions. Such scenarios necessitate appropriate solutions to the following important issues:

The credibility of observation data: collected ratings from various sources can either be trustworthy or biased depending on the inherent behaviors and motivation of the ones sharing the feedback. Thus the understanding and evaluation of malicious reporting behaviors need to be taken into consideration adequately (Dellarocas, 2005b; Despotovic & Aberer, 2006; Jøsang *et al.*, 2007).

The multi-dimensionality of quality and trust: QoS and trust are multi-faceted, inter-dependent, and subject to many other factors. This issue further complicates our estimations of peer service quality and peer rating behaviors, because we have to take into account various dependencies among quality parameters and related factors, whose values can only be observed indirectly via (manipulated) ratings. The incompleteness of the observation data set is an additional issue to be considered.

The subjectivity of quality and trust: the evaluation and perception of quality and trust information strongly depend on the view or the execution context of the evaluating user. For example, according to its personalized preferences, a peer may estimate the trustworthiness of another based on certain quality dimensions of the latter. Generally peers have different interpretations on the meaning of a trust value, therefore computing a globally-defined trust value for each peer may not be appropriate. A more suitable approach is to enable a peer to evaluate the well-defined quality attributes of the others based on collected experience, from which to personally evaluate the trustworthiness of its prospective partners.

The three above issues are *inter-related* and even *inseparable* for many application scenarios. In this perspective, I believe that more generalized results are still missing, as most research efforts are either ad-hoc in nature or only focus on specialized aspects. Many trust computational models in the literature either rely on various heuristics (Aberer & Despotovic, 2001; Xiong & Liu, 2004) or produce trust values with ambiguous meanings based on the transitivity of trust relationships (Kamvar *et al.*, 2003; Yu *et al.*, 2004). Other probabilistic-based trust evaluation approaches, e.g., (Buchegger & Boudec, 2004; Despotovic & Aberer, 2004b; Mui *et al.*, 2002; Patel *et al.*, 2005; Whitby *et al.*, 2005), are still of limited applications since they do not appropriately take into account the effects of contextual factors, the multi-dimensionality of trust, quality, and the relationships among participating agents[1].

[1] This statement holds at the time I started working on this framework. Later, I also found other probabilistic framework similar to mine that are developed independently somewhere else (Hsu *et al.*, 2006b; Regan *et al.*, 2006). Please refer to section 3.1.7 for a comparison of these approaches.

This solution is a probabilistic framework to support the subjective computation and personalized use of trust and quality information. This approach addresses the above questions adequately by offering several advantages:

- **Support personalized modeling of quality and trust:** using graphical models, our framework enables a peer to describe the quality and behaviors of the others flexibly according to personalized preferences. To the best of my knowledge, this work is among the first ones taking into account natural dependencies among QoS parameters, related contextual factors, and inherent behaviors of peers for the reliable estimation of trust and quality.
- **Offer a general decentralized approach to evaluate quality and trust:** this solution facilitates the use of various probabilistic learning and inference algorithms to evaluate quality and behaviors of peers. In the current implementation, I introduce the Expectation-Maximization (EM) and the Junction Tree (JTA) algorithms enabling a peer to estimate quality and behaviors of others locally and independently. These algorithms produce reliable estimates without using ad-hoc heuristics as in many other approaches. As a side effect, my framework can be seen as a *generalization* or a *theoretical basis* of many representative trust computational models in the literature, namely (Buchegger & Boudec, 2004; Despotovic & Aberer, 2004b; Mui *et al.*, 2002; Patel *et al.*, 2005; Wang & Vassileva, 2003; Wang *et al.*, 2006; Whitby *et al.*, 2005; Xiong & Liu, 2004).
- **Provide informative results with well-defined semantics:** since I use probabilistic graphical models to represent quality and trust, the trust computation produces outputs with clear and useful meanings, for example, it evaluates the probability that a peer is honest when reporting or the probability that another peer offers, for example, a data hosting service, with high download speed for a client with a certain type of Internet connection and willing to pay a certain price. This output can be used by the evaluating peer in many ways: (1) for its subjective trust evaluation, i.e., compute its trust on another according to various preferences, given the estimated values of different quality dimensions of the latter; (2) to choose the most appropriate service for execution given many functionally equivalent ones offered by the different peers in the system; and (3) to decide to go for interactions with other peers knowing their reporting behaviors, e.g., sharing and asking for experiences from them.
- **Have good performance and scalability:** the current implementation yields good performance even with reasonably high level of malicious rating users in the system. It is

also applicable and works well in case the collected multi-dimensional recommendations data set is incomplete. Additionally, my analysis and empirical experimentations have shown that the implemented framework is *scalable* in terms of performance, computation and communication cost.

In the next section I will describe our general framework for the personalized modeling and evaluation of trust and quality. Sections 3.1.3 and 3.1.4 present my implementation of this framework for the learning and inference of quality and trust. Section 3.1.5 presents the analytical and experimental results clarifying the advantages of my proposed approach. Section 3.1.7 summarizes the related work and Section 3.1.8 concludes my findings in this work.

3.1.1 Problem Description

Suppose that we are concerned with the values of the QoS parameters $\mathcal{Q} = \{q_i, 1 \leq i \leq m\}$, of a service s provided by a peer P in the system. Generally the value of q_i depends on many factors: other QoS attributes and certain environmental conditions. For example, given a file hosting service such as sendspace, megaupload, up-file.com, etc., the following QoS parameters are relevant: the offered download and upload speed, the time the server agrees to store the files, the allowed number of concurrent downloads, and so forth. The environmental factors that could affects those above quality attributes include: the price of the service, the location and the Internet connection speed of the user and so forth.

Whenever a new peer P_0 with no experience enters the system and wants to estimate the various quality properties $\mathcal{Q}$ of s, it needs to ask those peers j, where $1 \leq j \leq n$, who have used the service s of P for their experiences. Figure 3.1 shows such interactions where each j observes the quality level $q_{i_j} = u_j, q_{i_j} \in Q$ under the environmental conditions ϕ_j^*. Depending on its innate behavior and motivation, P_j reports the value $q_{i_j} = v_j$ as its perception on the quality parameter q_{i_j} of s (or generally P), where v_j may be different or the same as u_j. In this work I will assume that these reports from the peers js can be retrieved efficiently via appropriate routing mechanisms in the network, and peers have used available cryptography techniques, e.g., digital signatures, to ensure that these reports are authentic and can not be tampered with by unauthorized parties.

Suppose that we have collected many observations on various QoS parameters of P from several other peers. Each of these observation represents a (possibly biased) report on the quality of the target peer P under a certain environmental setting. Note that a peer j may

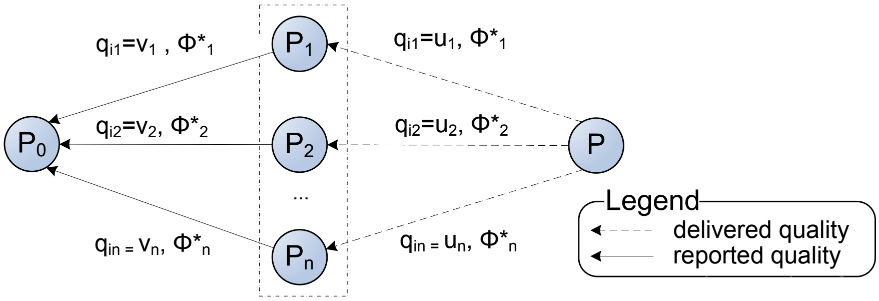


Figure 3.1: Sharing experience in a distributed setting.

submit several reports on different quality attributes v_j's to P_0. Given the above information, basically P_0 needs to estimate:

- the probability $p(q_i = c|\phi^*_{q_i})$ that the peer P offers q_i with quality level c under the environmental condition $\phi^*_{q_i}$ (or more generally the joint probability distribution of some quality parameters q_i);
- the probability $p(b_j)$ of the real behavioral model of a reporting peer j.

The answers to the above questions can be used for several purposes. For example, the output states whether the peer P performs better than another in terms of its QoS parameter q_i and under P_0's environmental settings, so that P_0 can select the more appropriate service to use. Also, given the estimated quality q_i's and based on its own preferences, P_0 can build its personalized trust on P flexibly. The evaluated behavior $p(b_j)$ of a peer j is also an indication of its trustworthiness and thus can be utilized by P_0 to decide whether to accept future interactions with j or not, e.g., for sharing and asking for experiences.

3.1.2 General Framework Architecture

My proposed framework (Fig. 3.2) has the following components to support the computation and personalized use of trust and quality information:

Modeling: since trust and quality are multi-faceted and subject to the personalized view of a user, the modeling of trust and quality is of high importance. Such modeling enables the personalization and reusability of the computational approach in many different application scenarios, given that the user can obtain necessary domain knowledge to formalize their problems appropriately. I propose the use of probabilistic graphical models, e.g., Bayesian networks, for the following important reasons:

- **Reality and generality:** such an approach models the reality elegantly: on one hand, the nature of quality and trust is probabilistic and dependent on various environmental settings; on the other hand, those dependencies can be easily obtained in most application domains, e.g., as causal relationships among the factors, which increase the reusability of our solution.
- **Personalized support:** user preferences and prior beliefs can be easily incorporated into a model, e.g., via the introduction of observed variables representing the behavior of trusted friends and hidden variables representing the quality and trustworthiness of unknown partners.
- **Tool availability:** algorithms to learn and inference on graphical models are widely available for implementation and analysis of the obtained results.

Let us take the file hosting scenario as an example. With certain helps from the user, a peer can identify most relevant QoS properties of such service (download, upload speed, maximal number of concurrent downloads, etc.) as well as related environmental factors (Internet connection speed, subscription price) to build an appropriate quality/trust model. These variables and dependencies among them can be easily obtained as causal relationships among corresponding factors in the domain, or even from the service description published by the provider peer.

Learning: given the quality model of the target peer, the evaluating peer selects a suitable learning algorithm to estimate the quality and trustworthiness of its target based on collected recommendations and own experience. Specifically, recommendations are ratings by other peers/users on certain quality dimensions of the peer being evaluated, e.g., the download and upload speed of its service. The selection of a learning algorithm usually depends on the constructed model, properties of the observation data set, and user preferences. For example, given an incomplete observation data set on a model with many hidden variables, the EM algorithm (Neal & Hinton, 1998) is a potential candidate. If the model is less complex and if the user prefer a full posterior distribution of the quality and behavior of the target peer, a Bayesian-based learning approach is more suitable. Subsequently, the computation of quality and trustworthiness of the target peer can be done by (probabilistically) inferencing on this learned model.

Decision making: the results of the learning process provide inputs for a user to evaluate his utility and choose the most appropriate actions, e.g., to select a file hosting provider or not. The actions of an agent may also lead to a refinement of its model on the others' quality and behaviors.

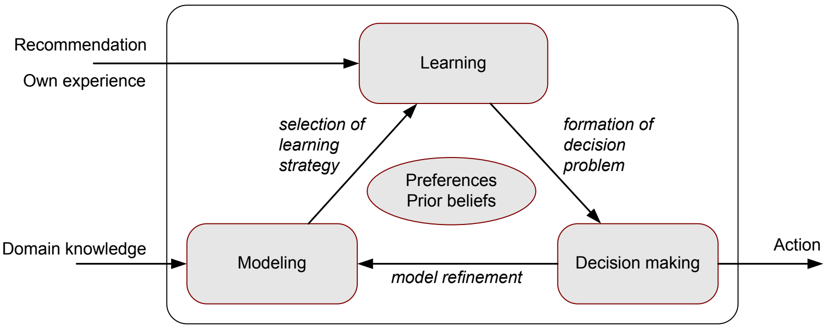


Figure 3.2: The probabilistic quality and trust computational framework

I implemented the above probabilistic framework with the following components. For the modeling step, I used directed acyclic graphical models, i.e., Bayesian networks, since I believe that they are sufficiently expressive to represent the causal relationships among various factors in our scenario, e.g., quality parameters, contextual variables, and peer behaviors. I apply the EM algorithm (Neal & Hinton, 1998) to learn the conditional probabilities of the constructed models and use the Junction Tree Algorithm (JTA) (Huang & Darwiche, 1996) to inference on them for the estimation of peer quality and behaviors. The trust-based decision-making step is not included in the current implementation, yet it can be incorporated easily.

3.1.3 Personalized Quality and Behavior Modeling

Denote $\mathcal{Q}$ the set of quality parameters of a service provided by a peer, where each $q \in \mathcal{Q}$ is assumed to have discrete values. While this assumption largely simplifies our problem and subsequent analysis, it is both realistic and advantageous. First, many quality attributes either have categorical values or are best represented with ranges of values due to their uncertain nature. Second, a rating on service quality is in fact the conformance between the quality values promised and delivered by the provider, as evaluated by the user. Such ratings are best modeled as discrete grading scales, as normally used in rating hotels or travel planning services. Similarly, let $\mathcal{E}$ be the set of contextual factors affecting values of quality parameters in $\mathcal{Q}$, where for simplicity only the case of contextual factors with discrete/categorical values are considered. The same methodology, however, applies in the case of continuous contextual factors and quality signals.

A peer (the learning peer) estimate quality of another peer (the target) by collecting and aggregating reports (ratings) on service quality of the target peer. Such reports come from a set $\mathcal{I} = \{j, 1 \leq j \leq n\}$ of other peers (previous users of the service) in the system. Among those rating peers, the learning peer trusts only some of them as reliable. Reliable (trusted) reporting peers $\mathcal{T} \subseteq \mathcal{I}$ may include, for example, those peers who provide commercial QoS monitoring services. Untrusted peers $\mathcal{I} \backslash \mathcal{T}$ are the set of previous service users, normally unknown to the learning peer. The set of trusted peers and those quality attributes to be monitored by them shall be private information of the learning peer. Otherwise, an adversary may disguise as an honest reporter to manipulate the QoS prediction of the learning peer effectively. For example, the adversary can provide reliable information only on those quality attributes monitored by trusted peers, while reporting biased feedback on the others.

Let $\mathcal{U}$ be the set of variables in a QoS generative model of a service provided by the target peer. A node $x \in \mathcal{U}$ represents either a quality attribute, a contextual factor, the trustworthiness of an information source, or the rating on a quality attribute. The domain value of x, or its state space, is denoted $\mathcal{D}_x$. A directed edge from one node to another denotes a probabilistic or causal dependency between two variables. For brevity, let us denote π_x the set of parent nodes of x. If x represents a quality attribute in $\mathcal{Q}$, the set of contextual factors in $\mathcal{E}$ that x directly depends on is $\phi_x = \pi_x \cap \mathcal{E}$.

The realization of x to some value $\mathsf{v} \in \mathcal{D}_x$ is denoted as $x = \mathsf{v}$ ($x \neq \mathsf{v}$ is defined similarly). Whenever it is irrelevant to mention v, we use the brief notation x^*. Similarly, for any given set $\mathcal{S}$ of random variables, we denote $\mathcal{S}^*$ the joint event in which each variable $s \in \mathcal{S}$ is assigned certain value in $\mathcal{D}_s$. The conditional probability that x gets some value given states of its parents π_x, or a *conditional probability table (CPT)* entry of the node x, is simply $Pr(x^* \mid \pi_x^*)$.

We then introduce the following important concepts.

Definition 1 *The QoS generative model of a service (by the target peer)* is a Bayesian network $\langle \mathcal{U}, \mathcal{D}, \theta \rangle$ with node set $\mathcal{U}$, edge set $\mathcal{D}$, and parameter θ. The tuple $\langle \mathcal{U}, \mathcal{D} \rangle$ is a directed acyclic graph where $\mathcal{D} = \{\langle p, x \rangle, x \in \mathcal{U}, p \in \pi_x\}$ defines probabilistic dependencies among nodes in $\mathcal{U}$. The parameter θ is the set of unknown *conditional probability table (CPT)* entries $Pr(x^* \mid \pi_x^*)$ of each node $x \in \mathcal{U}$.

Definition 2 *A QoS rating* by a peer $j \in \mathcal{I}$ at time t on a service is defined as $\langle j, t, \mathcal{V}_j^* \rangle$, where $\mathcal{V}_j^*$ denotes a set of reported values of j on a subset $\mathcal{V}_j \subseteq \mathcal{Q} \cup \mathcal{E}$. Specifically, $\mathcal{V}_j^* = \{x = \mathsf{r_{jt}}(x) \mid x \in \mathcal{V}_j\}$, where $\mathsf{r_{jt}}(x) \in \mathcal{D}_x$ is the reported value of j on x at time t.

In general a peer j may report only on a number of quality attributes or contextual factors $\mathcal{V}_j$. Thus values of several quality attributes and contextual factors may be missing in a QoS rating. To estimate the unknown parameter θ of a QoS generative model, related QoS ratings from many peers are combined to build a training set, which includes many *observations* (*samples*) on the model (Def. 3).

Definition 3 *An observation* or *a sample on a QoS generative model* is defined as $\mathbf{v}_\mu = \{x = \mathsf{x}^\mu, x \in \mathcal{O}\}$, where $\mathsf{x}^\mu \in \mathcal{D}_x$ is a rating value on a node $x \in \mathcal{O}$, and the set $\mathcal{O} \subseteq \mathcal{Q} \cup \mathcal{E}$ is a subset of quality attributes or contextual factors whose values during a measurement epoch μ can be obtained by combining ratings from certain peers. The set $\mathcal{O}$ maybe different for each sample.

Depending on its personalized preferences and prior beliefs on the outside world, an evaluating peer P_0 can model its view on the quality of a target peer P accordingly. We identify the following most typical models that a peer may use for various scenarios:

- **The basic QoS model** (Fig. 3.3a) is built on the assumption of the evaluating peer that all recommending peers are honest in giving feedback. Thus, those values reported by a peer are also its actual observations on the service quality. The nodes e_l, $1 \leq l \leq t$ represent those execution environmental (or contextual) factors influencing those quality attributes q_i, $1 \leq i \leq m$ of the service provided by peer P. This model is most useful in case the learning peer only collects rating from its trusted friends in the network.
- **The simplified QoS model** (Fig. 3.3b) extends the basic one to the case where P_0 believes that the collected recommendations are possibly biased. The node b denotes the reporting behavior of all recommenders, and v_i denotes their reported values on a quality attribute q_i. This model only considers the overall distribution of all peers' behaviors and is useful in case there are few reports on quality of the target peer P for the trust and quality evaluation step.
- **The extended QoS model** (Fig. 3.3c) extends the previous model to the case where P_0 believes that each reporter j may exhibit a different behavior b_j, $1 \leq j \leq n$. The variable v_{ij} denotes a rating value by a peer j on a quality attribute q_i of P. This extensive model is used if the learning peer P_0 also wants to evaluate the individual trustworthiness of each peer j in terms of giving recommendations.

A rounded square surrounding a node in Fig. 3.3 represents similar variables with the same dependencies with the others. Shaded nodes are variables whose values are observable and blank nodes are hidden (latent) variables to be learned from the collected reports. The numbers t, n, m are respectively the number of contextual factors, reporting peers, and QoS attributes that will

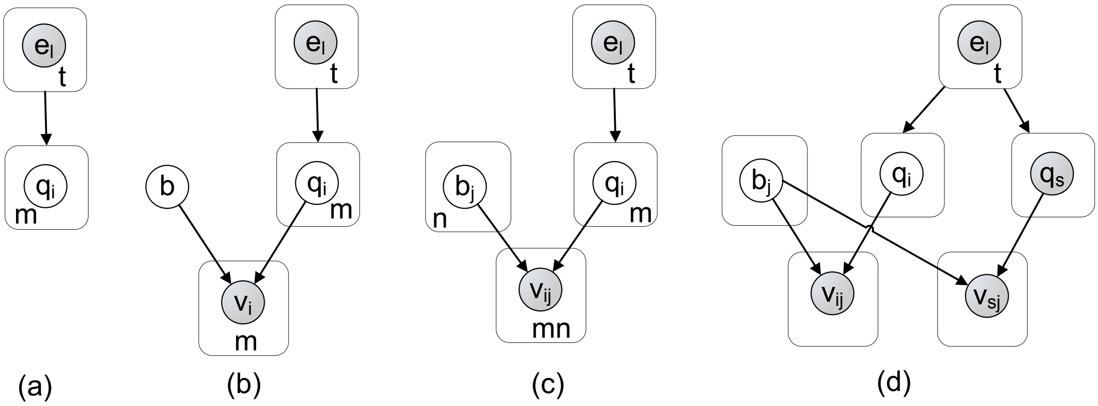


Figure 3.3: a. The basic QoS model of a service/peer; b. The simplified QoS model; c. The extended QoS model; d. Example personalized QoS model with some observed QoS properties q_s.

be used in later analysis. Using such probabilistic graphical models enable us to compute the probabilistic quality of a peer only by analyzing and traversing on corresponding graphs, thus widening the applicability of our approach.

A peer can include other prior beliefs and preferences to build its personalized QoS model based on the three typical models above. For example, let us assume that P_0 can collect correct observations on certain quality parameters q_s of the target peer P, either via its own experience or via the ratings of its trusted friends in the network. In that case P_0 can model those nodes q_s in the extended QoS model of P as visible to reduce the number of latent variables (Fig. 3.3d). There may also be dependencies among the different quality attributes q_i and among the nodes e_l, which are not shown for the clarity of presentation. The dynamic of various quality factors can be taken into account by considering time as a contextual variable in the models. Furthermore, it is possible to model the collusive groups in P_j by introducing further latent variables in the simplified and extended QoS models, but the learning will then be much more complicated. Consequently, in this chapter I will limit my analysis to the case where the evaluating peer collects random feedback from the network such that the behaviors of P_j can be assumed as approximately independent to each other.

3.1.3.1 Peer quality modeling example

Consider a system where peers are clients or providers of a data hosting service. The following quality attributes of a service are of our interests: its maximal number of concurrent downloads

M, its download speed D and its upload speed U. Values of these attributes are decided by the subscription price P and the Internet connection type I of the service consumer. In other words, P and I define the context, or the client-side setting in which the provider promises to offer its data hosting service with specific quality level of M, D, and U.

Fig. 3.4(a) shows a Bayesian network-based model of quality of the data hosting service with state spaces of all nodes and probabilistic dependencies among them. Such information is well-known in the data hosting domain and may even be declared by the provider in his service advertisement. Fig. 3.4(a), with conditional probabilities of each variable given state of parent nodes, is example of a *QoS generative model* (Def. 1) of the data hosting service provided by a peer in the network.

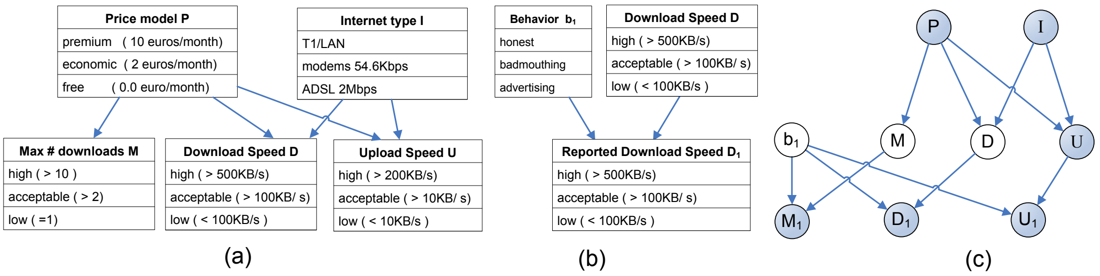


Figure 3.4: (a) Example quality attributes and related contextual factors of a data hosting service, state spaces of each node, and dependencies among them; (b) Dependencies among the trustworthiness (reporting behavior) of an untrusted feedback source and its corresponding reported values; (c) The QoS generative model of the data hosting service in simple form.

Suppose that feedback on the quality U is provided by a trusted peer, and feedback on M, D, and U is obtained from an untrusted peer with reporting behavior b_1. We use M_1, D_1, and U_1 to denote reported values by the untrusted source on M, D, and U, respectively. P, I, U, M_1, D_1, and U_1 are marked observable. Fig. 3.4(b) shows possible dependencies between the trustworthiness b_1 of a source and reported values U_1 on the download speed of the data hosting service. A feedback source observing a certain quality level may either report the same value, or deliberately give higher or lower rating value. E.g., a peer may give a bad rating on service of its competitors irrespective to the quality it actually perceives. These reporting behaviors are denoted as *reliable*, *advertising*, and *badmouthing* respectively, i.e., the feedback source is reliable with some unknown probability. Fig. 3.4(c) shows the QoS generative model after the second modeling step.

3.1.3.2 Using domain expert knowledge

Well-known constraints among values of variable nodes can be exploited to define certain CPT entries of a QoS generative model in Fig. 3.3. Particularly, one may set $Pr(v_{ij} = \mathsf{v} \mid b_i = \mathsf{reliable}, q_j = \mathsf{v}) = 1$, and $Pr(v_{ij} = \mathsf{v} \mid b_i \neq \mathsf{reliable}, q_j = \mathsf{v}) = 0$ for any $q_i \in \mathcal{Q}, \mathsf{v} \in \mathcal{D}_{q_i}, j \in \mathcal{I} \backslash \mathcal{T}$. Other value constraints of quality and contextual attributes can be exploited to set CPT entries of related nodes. Formally, a constraint may be of the form $\bigwedge_{x \in \mathcal{X}} x^* \rightarrow y^*$ or $\bigwedge_{x \in \mathcal{X}} x^* \rightarrow (y \neq \mathsf{v})$, for some $\mathcal{X} \subseteq \mathcal{U}$, some $y \in \mathcal{U}$, and some $\mathsf{v} \in \mathcal{D}_y$.

Define $\mathcal{KB}$ be the domain knowledge built from those constraints. For each $x \in \mathcal{U}$, the following rules apply: if $\mathcal{KB} \models \{\pi_x^* \rightarrow x^*\}$, we define $Pr(x^* \mid \pi_x^*) = 1$[1]. If $\mathcal{KB} \models \{\pi_x^* \rightarrow (x \neq \mathsf{v})\}$, set $Pr(x = \mathsf{v} \mid \pi_x^*) = 0$. Prolog programs can help to analyze such complex constraints to set values of related CPT entries.

As an illustration, consider the basic QoS generative model for the previous data hosting service of a peer (Fig. 3.4a). We can exploit the following constraints to define some CPT entries of the model. Given the above reporting behavior model, for any $\mathsf{u} \in \mathcal{D}_U$ we have $Pr(U_1 = \mathsf{u} \mid b_1 = \mathsf{reliable}, U = \mathsf{u}) = 1.0$, $Pr(U_1 = \mathsf{high} \mid b_1 = \mathsf{advertising}, U = \mathsf{u}) = 1.0$, and $Pr(U_1 = \mathsf{low} \mid b_1 = \mathsf{badmouthing}, U = \mathsf{u}) = 1.0$. Also, if the maximal number of downloads is deterministically defined by the price, we set $Pr(M = \mathsf{high} \mid P = \mathsf{premium}, I^*) = 1$, $Pr(M = \mathsf{acceptable} \mid P = \mathsf{economic}, I^*) = 1$, and $Pr(M = \mathsf{low} \mid P = \mathsf{free}, I^*) = 1$. Thus after the third modeling step we have an extended QoS generative model for the data hosting service with certain known CPT entries. The unknown parameter θ of the model consists of the remaining unknown CPT entries of b_1, M, and D.

Many existing works use certain instances of the above generative models for their QoS prediction, though the modeling step is not stated explicitly. For example, those approaches considering a number of independent quality attributes (Liu *et al.*, 2004) consider a generative model of independent nodes q_i. Some other work assuming the total reliability of feedback sources, such as (Maximilien & Singh, 2005; Shao *et al.*, 2007), and simply use the basic QoS generative model and propose a training algorithm based on some heuristics to estimate the model parameter θ. As a result, this work provides a framework to apply various probabilistic machine learning techniques to evaluate service quality which generalizes existing approaches.

The above steps result in a (personalized) QoS generative model of the service provided by the target peer, with some pre-defined CPT entries. Normally, in a given application, such a QoS

[1] We use the notation $\models$ to represent the deduction (provability) of a fact from a knowledge-base in a Hilbert-style deduction system

generative model is set up once for each service type, with possible help of domain experts. It is also possible to include dependencies among quality attributes q_i, or among contextual factors e_l in Fig. 3.3 according to the domain knowledge into the quality modeling and subsequent learning steps. These dependencies are not shown for the presentation clarity. However, my proposed algorithms in coming sections are generic enough to handle these cases. Also, I assume that values of contextual factors are verifiable and thus are observable variables. This is not a strong assumption since there is no direct incentive for feedback sources to manipulate such values, but only values of concerned quality parameters. Nevertheless, whenever contextual values are subject to manipulation, it is trivial to extend the above models and use the same approach with contextual factors marked as hidden nodes. It is also noteworthy that available domain constraints are very useful as they help to define many conditional probabilities of the model and thus reduce the size of the parameter set θ to be estimated in later steps.

3.1.4 Learning of Quality and Trust

Before using the QoS generative model for the estimation of service quality of a peer, we need to train the model to learn its unknown parameter θ. Specifically, collected QoS ratings from many sources (reporting peers) are used as training data to update and estimate remaining unknown CPT entries of the model.

3.1.4.1 Training data preparation:

A training data set includes many samples on the QoS generative model, each sample is obtained by combining related QoS ratings from different peers during a measurement epoch (Def. 3).

As in the modeling, the knowledge base $\mathcal{KB}$ built from domain constraints can also be used to filter out impossible observations in the report data set. For example, if we know that in the data hosting service domain, the maximum concurrent number of downloads for a subscribed user must be greater than 1, it follows that $Pr(M = \mathsf{low} \mid P \neq \mathsf{free}) = 0$. Thus any report by a peer j of the form $\{M = \mathsf{low}, P = \mathsf{economic}\}$ or $\{M = \mathsf{low}, P = \mathsf{premium}\}$ is impossible and should be filtered out.

Consider a QoS rating $\langle j, t, \{x = \mathsf{r_{jt}}(x) \mid x \in \mathcal{V}_j\}\rangle$ submitted by a peer j (see Def. 2). This report is removed if it violates certain domain constraints in $\mathcal{KB}$. Such a violation is formally equivalent to:

$$\exists x : \mathcal{KB}, \bigcap_{y \in V_j, y \neq x} \{y = \mathsf{r_{jt}}(y)\} \models \{x \neq \mathsf{r_{jt}}(x)\} \tag{3.1}$$

Violations as (3.1) can be detected by Prolog programs to eliminate invalid ratings of a source j. As j is likely to be unreliable, its related nodes in Fig. 3.3b. $(b_j, v_{ij} : q_i \in \mathcal{Q})$ are also eliminated. This preprocessing reduces the learning cost in many ways: it reduces the size of the training data set, while preserving most relevant information for the learning phase. Also, many nodes related to unreliable sources are eliminated, thus simplifying the model and reducing the number of parameters to be estimated.

Ratings on quality of the target service are transformed into observations on the QoS generative model as in Algorithm 1. The function *selectRatingsInEpoch*$(\mu, \mathcal{R})$ returns those QoS ratings in the set $\mathcal{R}$ whose timestamps t are within a measurement epoch μ. The function *findReportedNode*(j, x) returns the node v that represents the reported value by j on the node x, which will be different depending on the type of the QoS generative model (c.f. Fig. 3.3).

Algorithm 1 *PrepareTrainingData*(*reportData* $\mathcal{R}$): *trainingData* $\mathfrak{T}$

```
for $\mu = 0$ to $NumMeasureEpochs$ do
  $\mathbf{v}_\mu = \emptyset$; $\mathcal{R}_\mu = selectRatingsInEpoch(\mu, \mathcal{R})$;
  for each report $\langle j, t, \{x = \mathsf{r}_{\mathsf{jt}}(x) \mid x \in \mathcal{V}_j\}\rangle$ in $\mathcal{R}_\mu$ do
    if $j \in \mathcal{T}$ { reports from a trusted source } then
      $\mathbf{v}_\mu = \mathbf{v}_\mu \cup \{x = \mathsf{r}_{\mathsf{jt}}(x)\}$;
    else
      for each $x \in \mathcal{V}_j$ do
        $v = findReportedNode(j, x)$;
        $\mathbf{v}_\mu = \mathbf{v}_\mu \cup \{v = \mathsf{r}_{\mathsf{jt}}(x)\}$;
      end for
    end if
  end for
  Add a sample $\mathbf{v}_\mu$ to $\mathfrak{T}$;
end for
```

3.1.4.2 Learning parameter of the QoS generative model

Given a training data set appropriately filtered, there are two well-known approaches for estimating the parameter θ of the QoS generative model:

- **Frequentist-based approach**: methods of this category estimate the model parameters such that they approximately maximize the likelihood of the observation data set, using different techniques such as Maximum Likelihood Estimation (MLE) or Expectation-Maximization (EM) algorithm. In this solution class, the EM algorithm appears to be a potential candidate in our case since it works well on a general QoS model whose log likelihood function is too complex to be optimized directly. In addition, the EM algorithm can deal with incomplete data and is shown to converge relatively rapidly to a (local) maximum of the log likelihood. However, the disadvantages of an EM-based approach are its possibility to reach to a sub-optimal result and its sensibility to the sparseness of the observation data set.

- **Bayesian method**: this approach considers parameters of the model as additional unobserved variables and computes a full posterior distribution over all nodes conditional upon observed data. The summation (or integration) over the unobserved variables then gives us estimated posterior distributions of the model parameters, which are more informative than the results of frequentist-based methods. Unfortunately, this approach is expensive and may lead to large and intractable Bayesian networks, especially if the original QoS model is complex and the observation data set is incomplete.

The current implementation of the framework use the frequentist-based approach, in particularly the MLE method and EM algorithm to learn the quality and behavior of peers encoded as unknown variables in a QoS model. This application of an EM algorithm in the implementation is mainly due to its generality and promisingly practical execution cost. The use of other learning methods in this framework, e.g., an approximate Bayesian learning algorithm, and compare them with an EM-based approach is subject to future work.

3.1.4.3 Maximum Likelihood Estimation (MLE) learning

The unknown parameter of a QoS generative model can be estimated with traditional MLE techniques. This applies when all nodes of the model are observable, e.g., the case of a basic QoS generative model, or if training samples do not contain missing values, e.g., every reporting peer rates on all quality attribute of the service being evaluated. Consider a simpler version of the example data hosting service with two QoS attributes D and U as shown in Fig. 3.5. Assume that feedback on D and U is collected from an untrusted source with behavior b_1, and feedback on U is also provided by a trusted source.

As explained before in section 3.1.3.1, for any value d of D, $Pr(D_1 = \mathsf{d} \mid b_1 = \mathsf{reliable}, D = \mathsf{d}) = 1, Pr(D_1 = \mathsf{high} \mid b_1 = \mathsf{advertising}, D = \mathsf{d}) = 1, Pr(D_1 = \mathsf{low} \mid b_1 = \mathsf{badmouthing}, D = \mathsf{d}) = 1$. Similarly, for any $\mathsf{u} \in \mathcal{D}_U$: $Pr(U_1 = \mathsf{u} \mid b_1 = \mathsf{reliable}, U = \mathsf{u}) = 1, Pr(U_1 = \mathsf{high} \mid b_1 = \mathsf{advertising}, U = \mathsf{u}) = 1, Pr(U_1 = \mathsf{low} \mid b_1 = \mathsf{badmouthing}, U = \mathsf{u}) = 1$. Unknown CPT entries of the model in Fig. 3.5 are: $Pr(b_1 = \mathsf{reliable}) = h, Pr(b_1 = \mathsf{advertising}) = a$, $Pr(D = \mathsf{high}) = x, Pr(D = \mathsf{low}) = m$, and $Pr(U = \mathsf{high}) = p, Pr(U = \mathsf{low}) = q$. The unknown parameter of the model is thus $\theta = \{h, a, x, m, p, q\}$.

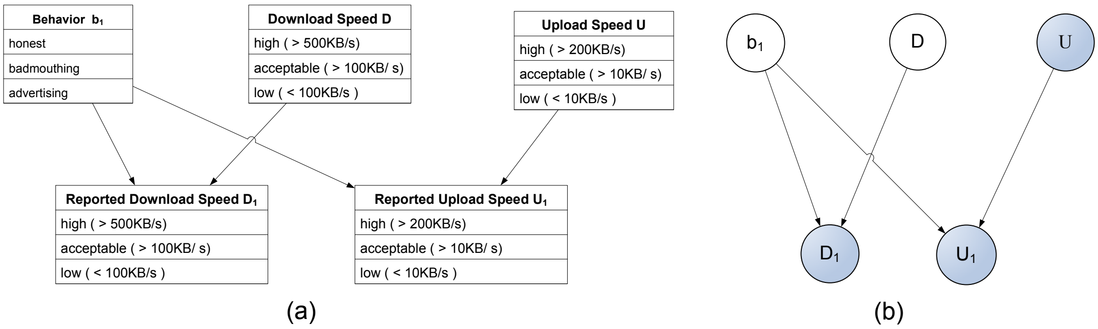


Figure 3.5: The generative QoS model of a data hosting service with two QoS parameters D and U, monitored by two group of reporting peers, one of which is untrusted with behavior b_1.

Suppose that we have a training data set $\mathfrak{T} = \{\mathbf{v}_\mu, 1 \leq \mu \leq N\}$ from feedback on D, U built according to Algorithm 1. Each observation $\mathbf{v}_\mu$ is the combination of ratings from the trusted source (on U) and the untrusted source (on D and U), so $\mathbf{v}_\mu = \{U = \mathsf{u}^\mu, U_1 = \mathsf{u}_1^\mu, D_1 = \mathsf{d}_1^\mu\}$. The probability of getting an observation $\mathbf{v}_\mu$ is:

$$\begin{aligned} Pr(\mathbf{v}_\mu \mid \theta) &= Pr(u^\mu, u_1^\mu, d_1^\mu \mid \theta) \\ &= \sum_{b_1, D} Pr(b_1)Pr(D)Pr(u^\mu)Pr(d_1^\mu \mid b_1, D)Pr(u_1^\mu \mid b_1, U) \end{aligned}$$

We select θ maximizing the log likelihood $LL(\theta)$ of obtaining the training set $\mathfrak{T}$, i.e., $\theta = \mathsf{argmax}_\theta LL(\theta) = \sum_\mu Pr(\mathbf{v}_\mu \mid \theta)$. Basic transformations give us[1]:

$$\begin{aligned} LL(\theta) &= \sum_{\mu:u^\mu=\mathsf{high}} \log p + \sum_{\mu:u^\mu=\mathsf{low}} \log q + \sum_{\mu:u^\mu=\mathsf{acceptable}} \log(1-p-q) \\ &+ \sum_{\mu:d_1^\mu=\mathsf{high}} \log[1_{\{u_1=u_1^\mu\}} xh + 1_{\{u_1^\mu=\mathsf{high}\}} a] \qquad (3.2) \\ &+ \sum_{\mu:d_1^\mu=\mathsf{low}} \log[1_{\{u_1=u_1^\mu\}} mh + 1_{\{u_1^\mu=\mathsf{low}\}} (1-a-h)] \qquad (3.3) \\ &+ \sum_{\mu:d_1^\mu=\mathsf{acceptable}} \log[1_{\{u_1=u_1^\mu\}} (1-x-m)h] \qquad (3.4) \end{aligned}$$

One may readily estimate θ from reported values $\mathsf{u}^\mu, \mathsf{u}_1^\mu, \mathsf{d}_1^\mu$ from (3.4). E.g., we have $p = \frac{\sum_\mu 1_{\{\mathsf{u}^\mu=\mathsf{high}\}}}{\sum_\mu 1}$ and $q = \frac{\sum_\mu 1_{\{\mathsf{u}^\mu=\mathsf{low}\}}}{\sum_\mu 1}$, similar to standard majority voting techniques for ratings on U. The reliability of the feedback source (h) can be estimated numerically via standard optimization techniques such as gradient descent. This results in an estimate of h based on

[1] The indicator function 1_A evaluates to 1 if A is true and 0 otherwise

similarity between ratings u^{μ} of the unknown source and ratings u_1^{μ} of the trusted one, similar to intuition.

3.1.4.4 Learning by EM algorithm:

For the case the target peer provides services with many quality attributes and contextual variables, the resulting QoS generative models are more complex. Additionally, the training data set is likely to contain many missing values since each source may report on only a number of quality attributes. Direction maximization of the log likelihood function $LL(\theta)$ as above becomes non-trivial and does not give a closed-form solution of the estimated parameter θ.

Many approaches for estimating the parameter θ under such situations are applicable (Buntine, 1996). In this paper, we propose the Expectation-Maximization (EM) algorithm (Neal & Hinton, 1998) to estimate θ for its many advantages. First, it works well on any QoS generative model and is especially useful if the log likelihood function is too complex to be directly optimized. Furthermore, it deals with incomplete data and converges rapidly. Its main disadvantages are the possibility to reach to a sub-optimal estimate (a local maximum of the log likelihood) with a bad initialization. However, we believe that it is possible to have an acceptably good initialization of θ to compensate these disadvantages, given personalized beliefs of the learning user on the behaviors of service providers and raters. The EM algorithm is given in Algorithm 2.

Algorithm 2 outlines the EM parameter learning on a general QoS graphical model with discrete variables, i.e., any model in Fig. 3.3. Algorithm 2 is run locally and independently by each learning peer in the system to evaluate the quality and behaviors of other peers, after the evaluating peer has constructed an appropriate QoS model of its targets and collected sufficient QoS ratings to build a training data set.

Algorithm 2 *LearnParameter*(*model* $\mathcal{M}$, *trainingData* $\mathfrak{T} = \{\mathbf{v}_\mu, 1 \leq \mu \leq N\}$)

```
1:  Init unknown $Pr(x^* \mid \pi_x^*)$ for each node $x$ of $\mathcal{M}$; /*Initialize $\theta$*/
2:  repeat
3:    for each observation $\mathbf{v}_\mu$ in $\mathfrak{T}$ do
4:      Compute $Pr(x^*, \pi_x^* \mid \mathbf{v}_\mu, \theta)$, $Pr(\pi_x^* \mid \mathbf{v}_\mu, \theta)$ for each node $x$;
5:    end for
6:    $E_{xpx} = \sum_\mu Pr(x^*, \pi_x^* \mid \mathbf{v}_\mu, \theta)$; $E_{px} = \sum_\mu Pr(\pi_x^* \mid \mathbf{v}_\mu, \theta)$;
7:    for each node $x$ do
8:      $Pr(x^* \mid \pi_x^*) = E_{xpx}/E_{px}$; /* Update $\theta$ */
9:    end for
10:   Recompute $LL(\theta) = \sum_\mu logPr(\mathbf{v}_\mu \mid \theta)$ with new $\theta$;
11: until convergence in $LL(\theta)$ or after a maximal number of iterations;
```

Line 1 of Algorithm 2 initializes unknown CPT entries of the generative model. Each conditional probability $Pr(x^* \mid \pi_x^*)$ can be initialized in many ways depending on available information: from prior beliefs of the user, as in the service advertisement by the provider, or as completely random. For the data hosting service, the learning user may assign $Pr(b_j)$ with his or her subjective belief on the trustworthiness of the source j providing the feedback. CPT entries of M, D, U in the data hosting service may be initialized as in the service advertisement. Lines $3-6$ implement the Expectation step of the EM algorithm, where the expected counts E_{xpx} and E_{px} of two joint events (x^*, π_x^*) and π_x^* are computed, given each observation $\mathbf{v}_\mu$ in collected reports and the current parameter set θ. Any exact or approximate probabilistic inference algorithm can be used to compute the posterior probabilities $Pr(\pi_x^* \mid \mathbf{v}_\mu, \theta)$ and $Pr(x^*, \pi_x^* \mid \mathbf{v}_\mu, \theta)$. The Junction Tree Algorithm (JTA) (Huang & Darwiche, 1996), is a good candidate as it produces exact results, works for all QoS generative models, and is still computationally scalable as shown in our later analysis.

The Maximization step of the EM algorithm is implemented in lines $7-9$ of Algorithm 2, therein we update the model parameters $Pr(x^* \mid \pi_x^*)$ such that they maximize the training data's likelihood, assuming that the expected counts computed in lines $3-6$ are correct. The two Expectation and Maximization steps are iterated till the convergence of the log likelihood $LL(\theta)$ of the training data set $\mathfrak{T}$, which gives us an approximation of unknown CPT entries $Pr(x^* \mid \pi_x^*)$ of the QoS generative model.

The set of observable variables in the QoS generative model may be changed after a user runs Algorithm 2, uses the service, and updates statistics of some quality attributes q_i with his own experience. Thus the learning of the model parameters is a reinforcement process with increasing accuracy over time, given availability of more training data to estimate a fewer number of unknown parameters.

We do not consider possible optimization techniques, e.g., adjust the weight of each sample $\mathbf{v}_\mu$ according its recency, to accelerate the learning convergence. The use of other parameter learning algorithms, e.g., a Bayesian learning method, is also possible. These issues are subject to future work and thus beyond the scope of this thesis.

Subjective computation of quality and trust

After the model parameter learning step, it is possible to compute automatically the following joint probability $Pr(\mathcal{D}^* \mid \Phi^*)$. Such a probability quantifies at which level the peer offers the service with a set $\mathcal{D} \subseteq \mathcal{Q}$ of quality attributes at a desired level $\mathcal{D}^* = \{q^*, q \in \mathcal{D}\}$, given the setting of a client $\Phi^* = \{e^* \mid e \in \Phi \subseteq \mathcal{E}\}$.

This probability implies whether a peer provide a better service than another in terms of the quality features $\mathcal{D}^*$ under environmental setting Φ^*. Thus, the result can be used for ranking and selection of appropriate services among functionally equivalent ones given their expected quality levels. The trustworthiness of an untrusted feedback source j (i.e., group of reporting peers), i.e., $Pr(b_j)$, is also of our interests, as this helps to select more reliable sources for future estimation, especially it is costly to obtain information from such feedback sources.

Given a generative QoS model with a known parameter θ, the computation of the above probabilities using probabilistic inference algorithms is straightforward (Huang & Darwiche, 1996) as follows. Values of contextual variables in Φ are defined according to setting of the user, then a probabilistic inference algorithm, such as JTA (Huang & Darwiche, 1996) is run on the model to compute the probability $Pr(\mathcal{D}^* \mid \Phi^*)$. Regarding the trustworthiness of feedback sources, all probabilities $Pr(b_j)$ are already computed during the parameter learning step.

For example, suppose that we want to estimate the probability that the service with the QoS generative model in Fig. 3.4(c) provides data hosting service with download speed level $D = \mathsf{high}$. Also assume that the user wants a free service. The probability $Pr(D = \mathsf{high} \mid P = \mathsf{free})$ can be done automatically with available probabilistic inference algorithm, given a QoS generative model with known CPT entries. Due to space limitation, we refer readers to (Huang & Darwiche, 1996) for a comprehensive tutorial on possible inference algorithms. An example use of this algorithm to inference on a QoS generative model is given in Appendix A. The most important aspect is that the whole above inference procedure can be done fully automatically with an acceptable computational cost (Section 3.1.5).

3.1.5 Analysis of the Approach

3.1.5.1 Computational cost

I provide in this section some theoretical results showing the possibilities of my approach. These results are for the candidate algorithms in this work and generally suitable to any service domain. Many optimizations, however, can be used to increase their effectiveness and efficiency.

We will estimate the most significant computational cost in the framework, which is of the parameter learning and probabilistic inferences on a QoS generative model. Consider a worst case scenario where each of m quality attributes has t contextual factors as parent nodes. There are n feedback sources being used, none of which are trusted. Suppose that every node has a k-ary state space. In a specific domain and for a certain service, t, m, and k are fixed values, known to the service user, and typically much smaller than n. The Bayesian network of such a generative model has a total of $t + m + n + mn = O(n)$ nodes. The number of unknown CPT entries $Pr(x^* \mid \pi_x^*)$ in the parameter set θ is at most $n_\theta = (k-1)t + (k-1)k^t m + (k-1)n + (k-1)k^2 mn = (k-1)(k^2 mn + n + k^t m + t) = O(n)$. The term $k-1$ is due to the normalization constraints as each variable has a k-ary state space.

Assume that a peer j sends N reports on some quality attributes of the target peer whose service quality model is given in Fig. 3.6(a). In fact, N approximates the number of measurement epochs, or the number of samples $\mathbb{N}$ obtained from Algorithm 1. The functions *selectRatingsInEpoch* and *findReportedNode* can be implemented with computational cost $O(1)$, e.g., with hash-based storage techniques.

Since the cost of three loops in lines $1, 3$,7 of Algorithm 1 are respectively $O(N), O(n)$, and $O(t+m)$, the computational cost of Algorithm 1 is $O(Nn(t+m)) = O(Nn)$. Even better, this step can be done off-line. The computation that needs to be done on-demand is the estimate of $Pr(\mathcal{D}^* \mid \Phi^*)$, which has a cost of $O(n)$ (Proposition 1).

Proposition 1 *The computational cost of one probabilistic inference on the worst-case QoS generative model using the Junction Tree Algorithm (Huang & Darwiche, 1996) is* $O(n)$.

Proof 1 *The result of the moralization and triangulation steps of the Bayesian network in Fig. 3.6(a) is shown in Fig. 3.6(b). The corresponding junction tree of this Bayesian network*

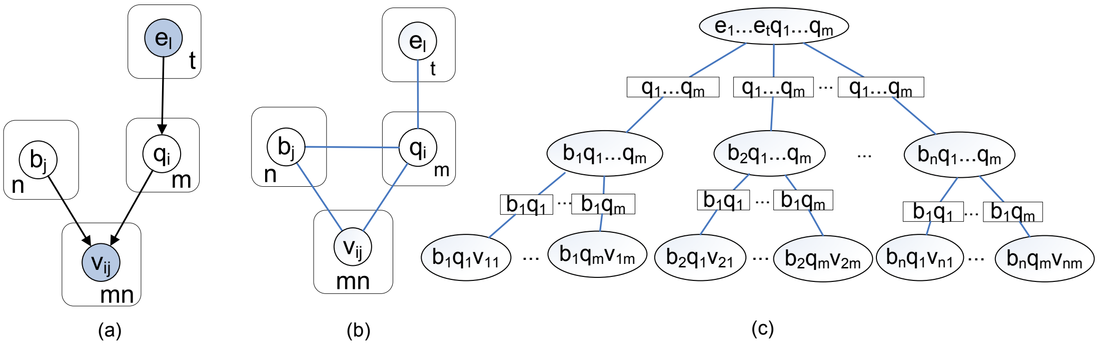


Figure 3.6: (a) The extended QoS generative model in worst-case scenario; (b) The extended QoS generative model after the moralization and triangulation step; (c) The junction tree of the extended QoS generative model.

is given in Figure 3.6(c). This junction tree has $nm + n$ edges and the maximal clique size with $t + m$ nodes. Thus the computational complexity of each inference using the JTA algorithm is $O(2(nm + n)2^{t+m}) = O(nm2^{t+m})$, where the factor $2(nm + n)$ corresponds to the number of messages passed during the inferences, and the complexity 2^{t+m} is the computational cost for marginalization of variables in the maximal clique $e_1 \dots e_t q_1 \dots q_m$ of the constructed junction tree. Therefore, one probability inference of the JTA algorithm cost $O(nm2^{t+m}) = O(n)$ for fixed (and usually small) t, m.

From Proposition 2, the cost of one EM iteration is $O(Nn^2)$. Given the fact that in practice the EM algorithm converges fast, and we consider a limited number of feedback sources, Algorithm 2 is scalable in terms of computational cost with respects to the training data size N.

Proposition 2 *The computational cost of one EM iteration of Algorithm 2 is $O(Nn^2)$.*

Proof 2 *From Proposition 1, the cost of computing the probability $Pr(x^* \mid C^*)$, where C^* denotes the setting of some variables in the generative model to certain states, is $O(n)$. This cost is higher than the cost of computing either $Pr(x^*, \pi_x^* \mid \mathbf{v}_\mu, \theta)$ or $Pr(\pi_x^* \mid \mathbf{v}_\mu, \theta)$ in line 4 of Algorithm 2, as the first requires an additional sum over all unwanted variables.*

Using the above result, one can verify that the computation cost for lines $3-6$ of Algorithm 2 is $O(Nn^2k) = O(Nn^2)$, with N is the number of observation cases. The computation cost for the loop in lines $7-9$ is $O(n)$, since there are $n_\theta = O(n)$ CPT entries $Pr(x^* \mid \pi_x^*)$ in the model.

The computation of the log likelihood $LL(\theta) = \sum_\mu log Pr(\mathbf{v}_\mu \mid \theta)$ (line 10 of Algorithm 2) involves the summation over all hidden variables $\mathbf{h}_\mu$ (variables whose values not reported) for each observation $\mathbf{v}_\mu$. Since the nodes $b_j, 1 \leq j \leq n$ are independent, each sum $Pr(\mathbf{v}_\mu \mid \theta)$ can be implemented with the cost $O(k^{t+m+1}n) = O(n)$ as follows:

$$Pr(\mathbf{v}_\mu \mid \theta) \quad = \quad \sum_{\mathbf{h}_\mu} Pr(\mathbf{h}_\mu, \mathbf{v}_\mu \mid \theta)$$

$$
\begin{aligned}
&= \sum_{\mathbf{h}_\mu \setminus \{b_1,\ldots,b_n\}} Pr(q_i^\mu \mid \pi_{q_i^\mu}, \theta) Pr(e_l^\mu \mid \pi_{e_l^\mu}, \theta) \\
&\times \prod_{j=1}^{n} \sum_{b_j} Pr(b_j \mid \theta) Pr(\mathbf{v}_{ij}^\mu \mid b_j, q_i^\mu, \theta)
\end{aligned}
$$

The terms $Pr(q_i^\mu \mid \pi_{q_i^\mu}, \theta)$, $Pr(e_l^\mu \mid \pi_{e_l^\mu}, \theta)$, $Pr(b_j \mid \theta)$, and $Pr(\mathbf{v}_{ij}^\mu \mid b_j, q_i^\mu, \theta)$ are known CPT entries of the current parameter set θ, which are already computed after the Maximization step (line $7-9$ of Algorithm 2).

Consequently, the total computation cost of each EM iteration in Algorithm 2 is $O(Nn^2) + O(n) + O(n) = O(Nn^2)$ for n contributing sources and N training samples.

3.1.5.2 Communication cost

The communication cost required by the learning peer P_0 is linearly dependent with the number of sources j $(1 \leq j \leq n)$ that provide ratings on the peer P. Assuming that each source j provides N reports on the quality of P, the total communication cost is $O(N \sum_{j=1}^{n} \tau_j)$, where τ_j is the cost for retrieving a report from j. Using a structured routing overlay, e.g., Chord (Stoica *et al.*, 2001), the complexity of τ_j is $O(log|V|)$, where $|V|$ denotes the number of all peers in the system. The complexity of the communication cost is therefore $O(Nnlog|V|)$, which mainly depends on the number of direct rating sources n. Thus one important question is to estimate n and N required for an acceptably accurate estimation of the quality of the target peer P. Note that n maybe much smaller than the actual number of rating peers: many of them maybe considered by the learning peer as having similar behaviors and thus can be encoded as one single rating source in the QoS generative model.

3.1.5.3 Learning errors and sample complexity

The sample complexity to learn the parameter of a fixed-structure Bayesian network, i.e., the asymptotic bound required number of samples $\mathbb{N}$ to learn θ with optimal error, is well-known (Buntine, 1996; Dasgupta, 1997; Wocjan *et al.*, 2002). In fact, the required number of samples in our framework is exponential in terms of the in-degree bound of nodes and grows less than linearly with the network size (Dasgupta, 1997; Wocjan *et al.*, 2002). For the QoS generative model in our worst-case scenario with a bound of node's in-degree of $O(t+m)$ and network size of $O(n)$, the sample complexity is $\mathbb{N} = O(n2^{t+m})$. This complexity implies that our approach is feasible for estimating QoS of services with a small number $t+m$ of quality attributes and contextual factors.

3.1.6 Experimental Results

Experimental setting

I implemented the presented computational framework using the BNT toolbox (Murphy, 2007) and tested it extensively under various conditions. Experiments were performed with the aforementioned file hosting scenario to study the accuracy of the estimated values under various practical settings: (1) the evaluating peer uses different ways to model the service quality of its target, e.g., using the basic, simplified, and extended QoS models; (2) there are different reporting behaviors; and (3) the recommendation set is incomplete and sparse. On one hand, the conditional probabilities of the QoS parameters of the service were set according to some (small) deviations from advertised QoS values to simulate possible cheating behavior of a provider peer. On the other hand, the observation data set was generated as samples on the visible variables of the QoS model according to various reporting behaviors of the service users, i.e., the reported quality values have different probabilities. A fraction of reported values were further hidden to create an incomplete recommendation data set.

The cost and performance of the algorithm are measured for the three cases where the learning peer uses the basic model M_b, the simplified model M_s, and the extended one M for its subjective evaluation of quality and trust on the others. In the most complicated setting with the extended QoS graphical model, the number of nodes in the probabilistic network is $2 + 3 + n + 3n$, in which the two first visible nodes denote the two environmental factors, the three next hidden nodes represents the three QoS attributes whose values depend on two previous environmental variables. There are n latent nodes representing the behaviors of n peers and $3n$ observable nodes storing the reported values of those n peers on three invisible QoS parameters. These $3n$ reported nodes are used to generate the various reports of several peers on the peer P with the service s being evaluated, which may also contain missing data and whose values are different to the actual QoS values the peers observed due to either the uncertainty in observation or the dishonesty of reporters.

The EM algorithm is initialized as follows: the conditional probability of each QoS parameter is set randomly and behaviors of all reporting peers are unknown. From the model with parameters learned by the EM algorithm, I estimated the distributions of the reporting behaviors of the peers and the QoS features of the service. I deliberately used the above setting since it is of the worst case scenario where the evaluating peer has no information of any other peer and no trusted friends. The experiments in other cases where the evaluating peer has different preferences and certain trusted friends are subject to our future work. However, I believe that even better results could be obtained in those cases, since there would have been more evidences to learn on models with fewer hidden variables.

The accuracy of the approach is measured as the *normalized square root error* of the quality and behavior estimations. The range of the normalized square root errors is $[0.0, 1.0]$ in which lower values implies higher estimation accuracy and vice versa. The experimental settings and corresponding results are summarized as in Table 3.1. Each experiment type consists of 20 experiments, each of which is averaged over 20 run times. These experimentations give us the following conclusions:

Computational cost scalability: Figure 3.7 shows the run time cost (in seconds) and the number of EM iterations used by the algorithm vs. the number of reporting peers for the three different QoS graphical models used by the learning peer P_0. In case of the basic model, the cost is very low, since the evaluation of different quality values is performed on a Bayesian

Table 3.1: Summary of experimental settings and results. f_c, f_u represent the percentage of cheating and probabilistic reporting peers with varied behaviors. The column f_i denotes the percentage of missing data in the observation data set

Experiment goal	n	N	$f_c(\%)$	$f_u(\%)$	$f_i(\%)$	Results
Cost vs. # of raters	1 to 100	5	45	5	0	Fig. 3.7
Perf. vs. # of raters	1 to 100	5	45	5	0	Fig. 3.8
Perf. vs. # of reports	15	1 to 20	45	5	0	Fig. 3.9
Perf. vs. effective attacks	15	5	0 to 95	0	0	Fig. 3.10
Perf. vs. incomplete data	15	5	50	5	0 to 95	Fig. 3.11

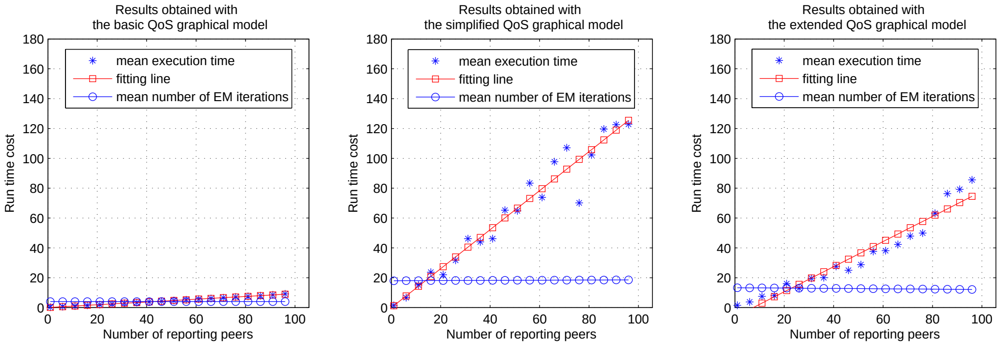


Figure 3.7: Run time cost vs. number of reporting peers.

network of fixed size and with no hidden nodes. That means the EM learning reduces to the MLE parameter estimation on fully observable Bayesian network, which is simple counting (Section 3.1.5.1) . For the simplified and the extended QoS models, the experimental results confirm that the computational cost is linearly proportional to number of reporting peers n, as previously shown in Section 3.1.5.1. We also observe that given a fixed number of reports N by each j, the number of EM iterations is bounded irrespectively of n. Note that with a given n, N, the number of samples for the simplified QoS model is larger than that of the extended QoS models, which explains the higher execution time of learning on simplified QoS model.

Performance scalability: generally the number of direct reporting peers n does not have any major influence in the performance of the algorithm (Fig. 3.8), but the number of observation cases does (Fig. 3.9). A peer only needs to get the reports from a certain number of other peers (chosen randomly) in the system to reduce the total learning time and communication cost without sacrificing much in the accuracy. Practically, in some other experiments, we only choose a set of $n = 15$ reporting peers randomly chosen in the network to reduce the total running time of the learning algorithm.

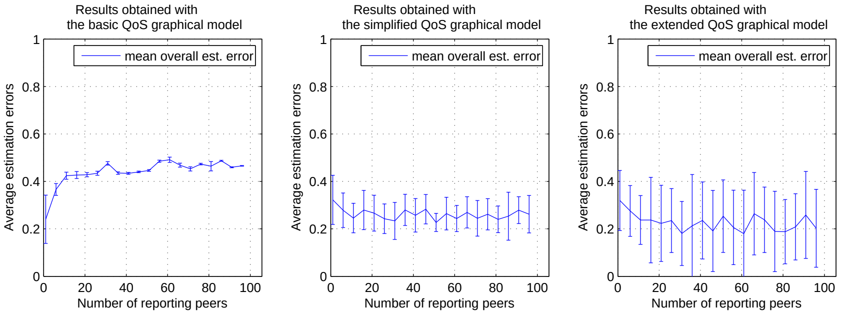


Figure 3.8: Estimation errors vs. the number of raters, $f_c = 45\%$ cheating raters.

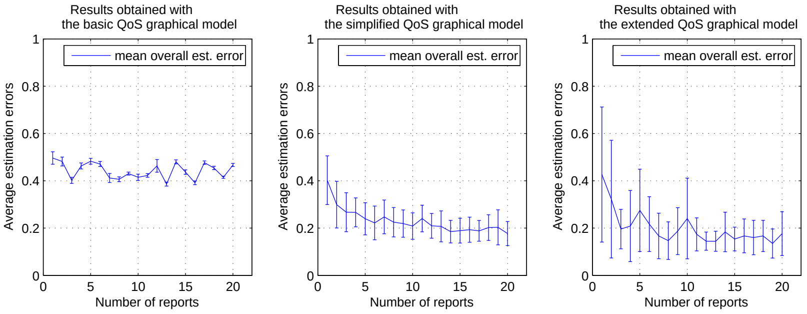


Figure 3.9: Estimation errors vs. the number of reports per peer, $f_c = 45\%$ cheating raters.

The importance of modeling (Fig. 3.8, 3.9, 3.10): the simplified QoS model yields best estimation results given only a few ($N = 3$ to 5) reports per peer (Fig. 3.9). Using the extended QoS model requires more reports to be collected for the learning step, otherwise its results are less accurate and more uncertain. This is reasonable since the extended model is much more complex and has many hidden variables to be learned. Thus simpler model is more preferable. The result of the basic QoS model is the least accurate and more vulnerable to malicious reports, since in this case the modeling peer trusts all raters completely. This observation confirms the necessity of an approach which enables the subjective modeling and evaluation of quality and trust. Depending on the availability of the reports from others, its prior beliefs on other peers and personalized preferences, a peer can choose an appropriate model for its personal estimations of quality and behaviors of prospective partners. To be specific, the performance of the estimation using the basic QoS generative model becomes worse and worse with increasing levels of cheaters in the reporting peers (see the leftmost of Fig. 3.10). The reason is that the basic model does not include the maliciousness of the reporting peers into any computation. On the other hand, using the simplified and extended QoS generative models, which explicitly consider the possible manipulation of ratings, the estimates are better and better with the increasing percentage of cheating raters, since these models become better fit to the reality (c.f. the middle and rightmost of Fig. 3.10). This observation suggests that a learning peer should select a model

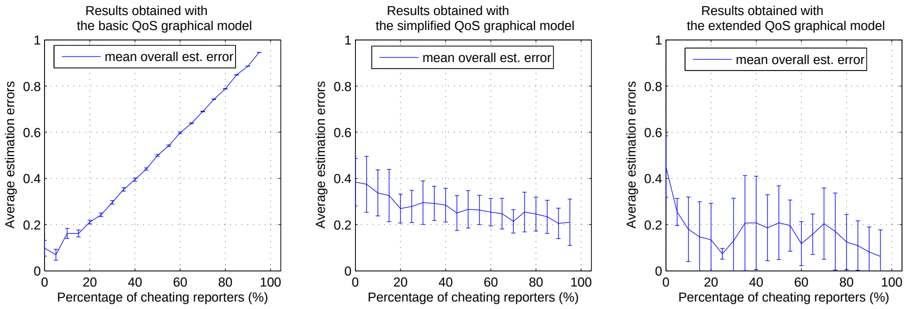


Figure 3.10: Estimation errors vs. cheating behaviors, , $f_c = 0\%$ to 95% cheating raters

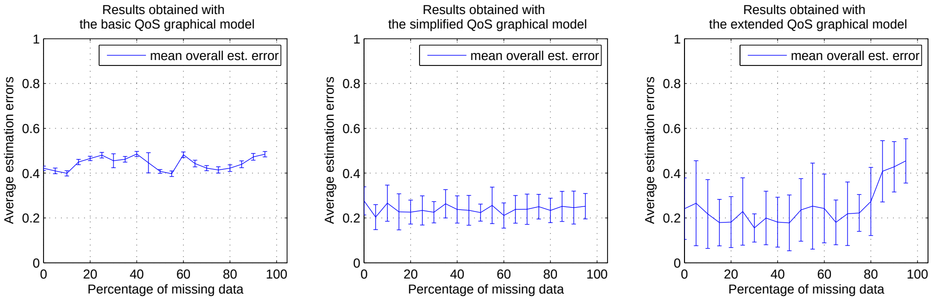


Figure 3.11: Estimation errors vs. percentage of missing data, $f_c = 50\%$ cheating raters.

to estimate performance of a target peer most fitted to the environment vulnerability. Such information can be even learnt from experience, e.g., by updating the model after making a decision. I conjecture that an adaptive learning approach to estimate the quality and trust, i.e., by revising the QoS generative model over time given more knowledge about the environment vulnerability, and by collecting of feedback from more reliable reporting peers, would have a much better performance. A more detailed evaluation of this approach, is subject to future work.

The effectiveness of the EM algorithm: the performance of the EM algorithm is tested extensively with respect to the following dimensions: (1) the average number of reports N submitted by each peer j; (2) the incompleteness of the observation data set; (3) and the percentage of honest reporting peers among js. Given enough observation data, generally using the EM learning algorithm on the simplified and the extended models can yield respectively good or acceptable estimation, even with a reasonably high number of malicious users and with an incomplete observation data set (Fig. 3.11). The result obtained via the use of the simplified QoS model is the most accurate, which is reasonable since this model also takes the various cheating behaviors of reporting peers into account appropriately. Using the extended QoS model gives us less accurate evaluations as the number of observation cases in this case is insufficient ($N = 5$) for learning on such a complex model. The result from of the basic

QoS graphical model is the less accurate, since this case means that the learning peer trust all reporting peers js completely.

3.1.7 Related Work

This work is mostly related to research efforts in development of reputation-based computational trust models surveyed in (Dellarocas, 2005a; Despotovic & Aberer, 2006; Jøsang *et al.*, 2007; Resnick *et al.*, 2000). The probabilistic framework presented in this chapter is different from the existing work in many aspects. Available approaches are based on various heuristics (Aberer & Despotovic, 2001; Xiong & Liu, 2004) or social network analysis techniques (Guha *et al.*, 2004; Kamvar *et al.*, 2003; Yu *et al.*, 2004). However, most of them are either ad-hoc in nature (Aberer & Despotovic, 2001), have high implementation overhead (Kamvar *et al.*, 2003) or do not produce estimates of trust with well-defined semantics (Xiong & Liu, 2004; Yu *et al.*, 2004). Other probabilistic-based trust evaluation approaches, such as (Buchegger & Boudec, 2004; Despotovic & Aberer, 2004b; Mui *et al.*, 2002; Patel *et al.*, 2005; Wang & Vassileva, 2003; Whitby *et al.*, 2005) do not appropriately take into considerations the effects of contextual factors and preferences of users, the multi-dimensionality of trust and quality, and the relationships among participating users.

This work, on the other hand, can be shown as the generalization of many existing approaches, e.g., (Despotovic & Aberer, 2004b; Hsu *et al.*, 2006a; Mui *et al.*, 2002; Patel *et al.*, 2005; Regan *et al.*, 2006; Wang & Vassileva, 2003; Wang *et al.*, 2006; Whitby *et al.*, 2005). For example, (Despotovic & Aberer, 2004b) is a special case of the solution presented in this chapter if the reports and quality are one-dimensional and the EM algorithm is applied on the simplified QoS model to find those model parameters best fitting to the observation data set. PeerTrust(Xiong & Liu, 2004) evaluates the quality and trust value of a peer given its ratings by other peers and credibility of individual raters, therefore it is similar to the learning on an extended QoS model in my framework, where the reliability of a rater is assumed to be depending on the similarity of its reports with the report of the learning peer. Yet, my solution is well-supported by the probabilistic learning and inferencing techniques and relies on no ad-hoc heuristics. The approaches in (Buchegger & Boudec, 2004; Mui *et al.*, 2002; Patel *et al.*, 2005; Whitby *et al.*, 2005) are similar to my computational framework under the assumption that user ratings and peer quality are one-dimensional, binary-valued, and the learning of peers' behaviors is done using a Bayesian-based approach to inference on the extended model given the observation data set. Comparing to the work in this chapter, (Wang & Vassileva, 2003; Wang *et al.*, 2006) do not take into account the dependencies among different quality dimensions of the peer as well as the relationships between the observed and reported values of peers in the system with respect to their behaviors for an accurate estimate of quality and trust. In (Regan *et al.*, 2006) the authors propose modeling e-market services as a Bayesian network and use a Bayesian learning approach to estimate distribution of the model parameters. (Hsu *et al.*, 2006a) uses a simple Bayesian network to learn the trustworthiness of a party by the EM algorithm. These approaches do not exploit domain knowledge on service structure and presence of trusted parties to reduce the cost of probabilistic learning and inference on the models. As a side-effect, my framework, though independently developed, appears to subsume these specific approaches.

3.1.8 Conclusion

I have proposed a general probabilistic framework to support the personalized computation of quality and trust in decentralized systems. This framework gives us several interesting findings. Most importantly, it is shown that modeling is very crucial in the design and implementation of a computational trust model, especially if trust is built on multi-dimensional qualitative behavior of agents, and where ratings are collected from information sources with unknown reliability. Domain knowledge and human experts are very important for the design of a good generative model for trust learning.

The first implementation of the framework using Bayesian network modeling, the EM and JTA algorithms have been shown to be scalable in terms of performance, run-time, and communication cost. Additionally, it exhibits several advantages: it works well given few and incomplete feedback data from the service users; it considers the dependencies among QoS attributes and various related factors to produce more accurate estimations; the computed results have well-defined semantics and are useful for many application purposes.

Additionally, many existing computational trust models are in fact built on different generative models of the user's behaviors, where trust evaluation is done using simple learning heuristics. Therefore, it is possible that similar approaches can be developed and better analyzed under this umbrella theoretical framework, e.g., by comparing which approaches yield optimal error in learning the model parameter via well-known machine learning performance metrics.

3.2 On Prototyping and Simulation of Decentralized Reputation-based Trust Systems

Although decentralized trust management has attracted substantial research efforts over the recent years, existing computational trust models mainly rely on simulation results to verify their performance and effectiveness. Such experiments are usually conducted in different ad-hoc simulation environments, making comparison of existing solutions extremely difficult.

Given this observation, another goal of this thesis is to provide a generic and powerful framework to assist researchers in their development and evaluation of novel trust management approaches while minimizing implementation efforts. Specifically, one only needs to focus on the development of the core algorithms, not on setting up of the simulation environment. Since there is no open and well-known accepted simulation and experimentation framework for trust-related research at the moment, this framework is an important contribution to the research community. The ultimate goal is to establish a standardized powerful simulation test-bed for the trust-related research communities. More specifically, such a simulation framework is expected to be useful for trust-related researchers in several aspects:

- as a **powerful experimentation platform** to enable empirical simulation to discover new trust and cooperation-related phenomena. For example, we may want to use this simulator to test the emergence of trust and cooperation in various scenarios where peers use different learning algorithms and with various behaviors, including honest, malicious and strategic.
- as a **competition test-bed with rich testing environment**. The testbed can be used with similar purpose as the ART test-bed (Fullam *et al.*, 2005) to test comparative performance of different trust learning algorithms. To this extent, the developed simulator should be more flexible than the ART test-bed, which only simulates a painting art appraiser scenarios. My goal is to design a generic simulation framework where different application scenarios can be set up, and simulation parameters can be defined more flexibly.
- as a **prototyping tool**: the basic building blocks provided by the developed simulation framework can also help one develops and tests performance of his or her trust learning and decision making algorithms under a variety of standard and/or customized testing scenarios with minimized development efforts. The implemented trust management algorithms can also be used as an application-independent library of reputation-based trust learning algorithms to be used in a specific scenario.

This simulation framework is used in two scenarios, each of which is a research work presented in detailed in Chapter 4 and Chapter 5.

3.2.1 Requirements of a Simulation Platform for Trust Management Research

For most research work focusing on testing performance of a computational trust model, testing with real systems is often an infeasible option. First, real data set is not usually available. Second, although many work obtained data set on real reputation and recommender systems to back up their experiments, it is always the case that more synthetic data need to be generated, e.g., to simulate an attack and to discover different vulnerabilities of the system. The reason is that we typically do not know the ground truth of data obtained from real reputation systems, e.g., the collected data set does not include the information on the genuine outcome of historical transactions or which rating values on which transactions are correct.

Reputation-based computational trust models are frequently used in dynamic and complex systems. This makes analytical solutions on these systems' performance are difficult to obtain without introducing many simplifying assumptions. For example, qualitative behaviors of services provided by peers are usually considered as one-dimensional signals, or that the system in homogeneous in terms of peer preferences. Consequently, the validity and/or applicability of such analytical approaches is rather limited. Therefore, whenever a more thorough study of the system is needed, simulations and experimental approaches are common practices. Simulation are good for testing complex analytically intractable models, and for verifying analytical result. Therefore, simulation models would need to be built on real world data set and observations in real systems.

Beside some common features of a simulation tool, such as the ease of use, the reproducibility of the results, good documentation and clean API code, I have identified following main requirements of a generalized trust simulation and prototyping platform for the trust-related research community.

- **Extensibility**: results shall be expressive and easy to process and to reproduce. The system must be design to facilitate the demonstration and simulation of various similar interaction scenarios with different degrees of centralization. The design shall maximize the reusability of code and minimize the efforts spent to develop a new simulation for another application scenario. For example, we can model peer to represent any active entities (agents) system stands for any open systems with different degrees of centralization. It is also expected that users can easily define new metrics and plug new algorithms into the system for evaluating without much effort.
- **Practicability**: the simulation model shall be built on many data sets and observations from real running systems. I try to obtain as much parameter distributions from real system as possible, e.g., the distribution of uptime and arrival time of peers, number of average transactions, content distribution. Those parameters that can not be observed directly (such as peer behaviors) can be drawn from similar networks/systems (trace-driven).
- **Scalability**: the system must support simulation of a large number of peers and can be decentralized to support large-scale simulation.

My design of the trust simulation framework are focused on satisfying the above requirements as much as possible.

3.2.2 Existing P2P and Trust Simulators

Much effort has been spent in the development of simulators of P2P systems, some of which are ProtoPeer (Galuba *et al.*, 2009), PeerSim[1], FADA[2], JXTA[3]. More complete review in this area can be found in existing surveys (Naicken *et al.*, 2006). Also related to this line of research is the development of experimentation environments for research on network attacks and defenses[4]. PACE (Suryanarayana *et al.*, 2006) presents a software architecture to incorporate trust management algorithms in to peer applications. Those are complementary to the work in this chapter.

The most extensive and representative work in this area is the PeerSim project by Hales et al. This work focuses on simulation of communication among peers rather than the social behaviors of participants, which is more important for trust management research. A restriction of this

[1] http://peersim.sourceforge.net/

[2] http://fada.sourceforge.net/p2psim/

[3] http://www.sun.com/software/jxta/

[4] http://www.truststc.org/testbeds.htm

work is it mainly supports simple cycle-based simulation model, whereas the more realistic event-based simulation has not been gained much attention.

There exits a number of tools for simulating reputation system and decentralized trust models in the literature. Most tools are designed to simulate a single application scenario and do not provide extensive simulation and development support.

The most widely known tools for comparing different trust learning models is the ART testbed (Fullam *et al.*, 2005). This testbed is designed for competition among rational agents in an art appraisal domain and has attracted participants of many researchers over the recent years. A limitation of this testbed is that it is only applicable to testing performance of different trust approaches in the well-defined art appraisal domain. Furthermore, it is more suitable for limited number of agents rather than for experimentation effectiveness of trust learning algorithms in larger-scale environments.

Other work on simulation of reputation-based trust systems are those simulators developed for special purpose of individual research work, namely the SuppWorld framework (Sabater, 2003b), (Schlosser *et al.*, 2005), or Liang and Shi (Liang & Shi, 2008).

The main difference of my simulation framework from other works is its extensive supports for development and experimentation of trust management approaches. This framework is more generic and application-independent: support intelligent decision making, behavior modeling, learning and inference on multidimensional, context-dependent trust models, etc. Practically, the implemented framework has many useful features and provides several supports for the development and testing of new trust management approaches. First, the framework is designed to help fast prototyping and simulation a reputation system in any application scenario, thus it is much more powerful. Specifically, beside the possibilities of plugging in different learning, decision making algorithms as supported by ART testbed., in my framework various scenarios can be setup, e.g., peers may join and leave dynamically, peers provide services with multi-dimensional quality attributes. Thus the presented framework serves as a generic simulation platform, which can be easily configure to empirically test a specific learning algorithm under various designed attack scenarios. At the same time, the simulator can also be configured to use as a testbed without major effort.

3.2.3 Basic Design Concepts

To develop a generic platform that is capable of simulate different application scenarios, I consider the generalized interaction scenario introduced in Section 1.2. The system consists of many peers who provide or obtain *resources* of different *prices* and *quality* to and from the others. This abstraction makes it possible to tweak our system to simulate a range of similar applications, e.g., a C2C trading scenario, resource provisioning in Grid systems, or a market of services. This scenario represents several different practical P2P applications where trust and reputation information play an important role and are used extensively to improve the cooperation in the system. The simulation platform is built with a number of fundamental application-independent concepts as follows.

First of all, ResourceDescriptor is an abstract view of a resource in an application, which represents any object being evaluated, e.g., a service in service-oriented systems, a contact in a social networking site, a resource in Grid environments. Each resource has a number of quality properties with certain dependencies among them. Note that even reputation and trustworthiness can be viewed as a quality parameter as well. This modeling helps the simulation framework applicable for a wide variety of application scenarios where peers provide services or resources of different types to each other.

A peer (Peer) is an autonomous agent participating and interacting with other peers in the system. Thus a peer may simulate a real user, an intelligent agent, or a computer in the network depending on the application we want to simulate. Each individual peer can be either a provider, requestor, or rater of certain resources (ResourceDescriptor) and exhibits different behaviors, e.g., honest, dishonest, or strategic, in different contexts (ReportingBehavior, ServingBehavior). Behaviors of peers in the system are defined and configure by the system-wide data structure PeerSociety. Peers are identified by a unique address and has many preferences (PeerPreference). Example preference information is: which trust model, learning or decision algorithms to be used, the prior beliefs the peer has about the environment vulnerability, etc.

A peer may issue different Requests of various types to search for resource, look up ratings on a certain resource, and ask for using a selected resource. Upon the receiving of a request, the recipient can send back corresponding RequestResponse depending on its own reporting or serving strategy.

TrustModel is a data structure representing how a peer models its notion of trust on a certain resource. Depending on the application, one can build a trust model based on the list of quality parameters of the associated resource. A TrustModelNode is a node in the trust model, which actually represents a quality parameter of a resource. This comes from my observation that in many (if not almost) system, the trust value on a resource is generally defined based on its certain quality parameters. A TrustModelNode takes a TrustModelNodeState value. Note that in a report on quality of a resource may consists of many quality values. Values of quality properties in an observation can be either provided or not depending on the rater's willingness.

To make the simulation platform generic, the trust management subsystem is design following the holistic approach presented in Section 3.1.2. Specifically, I design the TrustManagement layer as composition of three subcomponents TrustModeler, TrustLearner, and DecisionMaker. Such design helps generalize the operation of almost all personalized/local reputation/trust management models in the literature. Fig. 3.12 shows the structure of the TrustManagement subsystem. The role of each component is specified as follows:

- the TrustModeler component formalizes the notion of trust and reputation of an object (a peer or one of its resources) via appropriate modeling. E.g., it may define trust as a joint probability that a peer offers resource with certain quality values. Given a trust model, the reputation of the described resource can be evaluated accordingly.
- TrustLearner: an abstract of a local trust learning algorithm of a peer. A peer uses collected observations and its own experience to revise its trust model, e.g., to change the parameter of the model to fit well to its beliefs and preferences. The revised/learnt trust

model will be used to evaluate the trustworthiness of the object the model describes.

- DecisionMaker: this component supports a peer in its intelligent trust-based decision makings. For strategic peers, this comprises those algorithms of a peer to maximizing the it long-term utilities. For naive/honest/cooperative peers, this component may be some simple deterministic rules. Examples decisions to be made are:
  (i) how to select a best partner in a certain transaction to maximize the long-term utilities;
  (ii) whether to accept, ignore, or serve a certain resource request and with which level of effort;
  (iii) how to share the collected ratings and with whom, e.g., only share ratings with its neighbors, trusted friends, other members in the same group;
  (iv) which learning algorithm to choose for estimating the quality and trustworthiness of the potential partners.

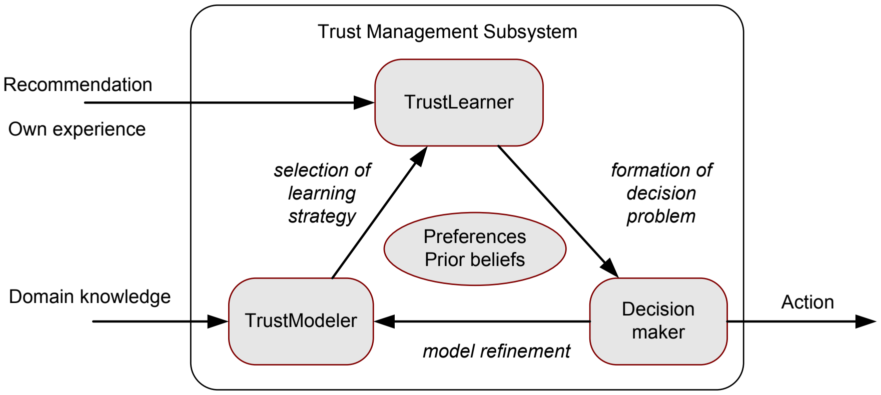


Figure 3.12: Details of the trust management subsystem in our prototyping and simulation framework. These subcomponents corresponding to the modeling, learning, and decision making steps presented earlier in Section 3.1.2.

3.2.4 Framework Architecture

Fig. 3.13 shows the overall architecture of the framework, which is composed of the following modular components. PeerApplication layer is an abstraction of the P2P application to be simulated. Such application is defined as an interaction scenario where peers provide or obtain resources of different prices and quality to and from the others.

The TrustManagement layer serves as a trust management subsystem supporting peers in modeling and learning trustworthiness of other targets, as well as supporting them in their trust-based decision making processes. Standard APIs and many basic building blocks are provided for users to develop and plug-in their own computational trust (learning) models and decision making algorithms into the system.

Another intermediate layer, the ServiceSupport layer encapsulates the distributed information storage, retrieval, and routing mechanisms being used. Particularly, this layer provides upper-layers with capabilities of sending requests and receiving responses among the peers, as well as searching and retrieving information on resources and related ratings on them for the trust computation. This layer separates the trust management subsystem from the mechanisms of identity handling, searching and retrieval of information. Thus the layer facilitates the integrating and testing of a certain trust management approach on top of different distributed information management mechanisms.

The SimulatedNetwork layer represents the underlying communication network, which can be implemented as a centralized or decentralized system depending on the system being simulated. Since we only emphasize on the simulation and study of social and strategic behavior of peers in the application layer for a certain trust management approach, only a basic implementation of this layer is provided. Specifically, the layer is implemented as a network of nodes with latencies among them following the King latency data set (Gummadi *et al.*, 2002), where peers join and leave the system according to a realistic churn model (Stutzbach & Rejaie, 2006). In fact, the system can be configured to use other message passing protocols or underlying communication networks with any latency models as needed, besides the default implementation.

The SimulationManager takes care of all back-end supports for the simulation, e.g., capturing, inspecting, controlling, and collection of measured data in both interactive and batch modes. RePastJ (North *et al.*, 2006) is used as the base to develop this component, as this library provides several useful tools for setting up the environment, dynamically controlling of simulation parameters, result visualization and analysis, etc.

Thanks to the modular designed and well-specified API of the subcomponents, each layers can be replaced by different implementations (simulated or real) according to the experimentation requirements of the application.

For example, it is possible to develop a decentralized implementation of the SimulationManager layer layer to support larger-scale simulations or even to *emulate* an application scenario on a real network, given participation of real users. Similarly, though implementation of the service support and communication layers are also provided for the completeness of the simulation framework, they can be easily replaced with current peer-to-peer simulation tools such as PeerSim[1].

3.2.5 Formal Models of the Simulation Framework

To have a better understanding of the operation of the presented simulation framework, I provide in this section a formal model of those concepts used above in the framework.

The semantic of trust and reputation in my simulator is interpreted as follows. Reputation of a resource is the *personalized belief* (e.g., subjective probability) of getting certain quality

[1] http://peersim.sourceforge.net/

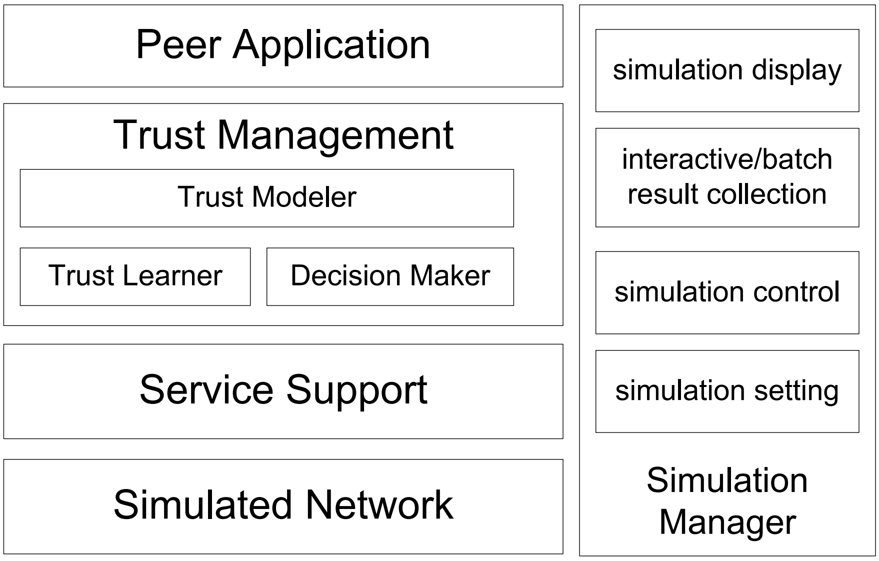


Figure 3.13: Architecture of the trust prototyping and simulation framework

under a known context and considered as past performance of the objects being evaluated. This belief is initialized with the learner's prior belief and revised with under the presence of personal experience and ratings from the others. The trust level on a resource is the prediction performance of a peer, e.g., the belief that a peer will offer a resource with desired quality level under a particular context, given the posterior belief and assumptions of the learner on the environment.

This interpretation helps us to build the following formal models of different concepts in the simulation framework. Let A be the set of all peers (agents) in the system who provide a set of resources (or services) S to each other. For a given resource $s \in S$, denote Q_s the set of all of its quality attributes q, each taking values in a domain D_q. Similarly, let C_s be the set of related contextual factors c, each taking values from a domain D_c. Let $\pi(x) \subseteq Q_s \cup C_s$ be the set of quality attributes and contextual factors that may influence value of a certain quality/contextual variable $x \in Q_s \cup C_s$. The domain D_q and D_c are application- and system-dependent, e.g., D_q can be a binary set $\{0, 1\}$ or a subset of real values $\mathbb{R}$.

3.2.5.1 Resource Modeling

Consider a resource $s \in S$ with the set of quality attributes Q_s and the set of related contextual factors C_s.

A *quality model* $\mathcal{M}_s$ of this resource is a directed graph $\langle Q_s \cup C_s, DE_s \rangle$ with the directed edge set $DE_s = \{(q', q) \mid q \in Q_s \cup C_s, q' \in \pi(q)\}$.

The *advertised quality level* $\mathcal{L}_s$ of this resource, as claimed by its provider is $\mathcal{A}_s = \{\ell_q \mid q \in Q_s\}$. Each mapping $\ell_q : \Pi_{x \in \pi(q)} D_x \rightarrow D_q$ corresponding to a claim of the provider to provide this resource with a specific level of quality attribute q under a defined context $\pi(q)$. This advertise quality $\mathcal{A}_s$ can be defined explicitly by the provider, e.g., as part of its service level agreement (SLA) description, or implicitly, e.g., ℓ_q is understood as having good values in the domain D_q irrespectively of the context.

A resource $s \in S$ provided by a peer p is formally defined by its *resource descriptor* $\mathcal{R}_s = \langle p, \mathcal{M}_s, \mathcal{L}_s \rangle$.

3.2.5.2 Trust Modeling

Suppose we have a resource descriptor $\mathcal{R}_s = \langle j, \mathcal{M}_s, \mathcal{L}_s \rangle$. A potential consumer peer i defines its *personalized trust model* $\mathcal{T}_s^i$ on that particular resource s as $\mathcal{T}_s^i = \langle \mathcal{M}_s, \mathcal{D}_s \rangle$, where $\mathcal{D}_s = \{\delta_q \mid q \in Q_s\}$ is the *expected-to-deliver quality level* of a resource s. Each mapping $\delta_q : \Pi_{x \in \pi(q)} D_x \rightarrow D_q$ corresponds to the personalized belief of peer i on the fact that the resource s will be provided with a specific level of quality attribute q under a defined context $\pi(q)$.

In general, this expected-to-deliver quality level $\mathcal{D}_s$ is different from those defined in its advertised quality level $\mathcal{A}_s$ and thus shall be learned by the modeling peer i. Such definition comes from our view that in many (if not almost) application scenarios, the trustworthiness of a resource is mainly defined based on its certain quality parameters.

A model $\mathcal{M}_s^i$ can also be encoded as a Bayesian network relating the various quality parameters of the resource (as in the probabilistic framework of Section 3.1.2). Alternatively, it can also be a visual representation of an ontology specifying relationships among different qualitative facets of the service/resource being described.

3.2.5.3 Peer Interactions

I adapt the formalism defined in (Schlosser *et al.*, 2005) to describe the simulation scenario. The time domain is discretized to $T = \{1, \ldots, W\}$, where W is the window-size of observations to be used by various peers.

Define E the set of all transaction (or encounters) between peers until now. A *transaction* $e \in E$ is denoted as a 5-tuple $\langle i, j, s, C, t \rangle$, where $i \in A$ is the resource consumer, j is the provider, s is the resource being provided, $C \in \Pi_{x \in C_s} D_x$ defines the context of the transaction, $t \in T$ is the associated transaction timestamp.

The set of quality values the resource consumer i observes after the transaction e constitutes an *observation* $o_e = \{v_q \in D_q \mid q \in Q_s\}$ on that transaction.

A *rating* (or a *report*) of a peer i on a transaction e is defined as $\rho(i, e)$, where $\rho : A \times E \rightarrow \Pi_{q \in Q^e} D_q$. The set $Q^e \subseteq Q_s$ includes those quality attributes of the resource s rated by the consumer i.

For later uses, we define S_j the set of resources provided by peer j. Also, we define E_j the set of transactions in which j is a resource provider and $E_{i,j} \subseteq E_j$ the set of all transactions between a consumer peer i and a provider peer j. Formally, $E_j = \{e \in E \mid e = \langle i, j, s, c, t \rangle, i \in A, s \in S_j, c \in C, t \in T\}$ and $E_{i,j} = \{e \in E \mid e = \langle i, j, s, c, t \rangle, s \in S_j, c \in C, t \in T\}$.

3.2.5.4 Trust Learning Algorithm

Suppose that we have a resource s with descriptor $\mathcal{R} = \langle j, \mathcal{M}, \mathcal{L} \rangle$. Let $\overline{\mathcal{T}}^i = \langle \mathcal{M}, \overline{\mathcal{D}} \rangle$ be the initial trust model of s formulated by the peer i. Normally, we have $\overline{\mathcal{D}} \equiv \mathcal{L}$.

A (personalized and reputation-based) *trust learning algorithm TA* takes input values $\langle G, \Omega, \overline{\mathcal{T}}^i \rangle$ and outputs an updated trust model $\mathcal{T}^i = \langle \mathcal{M}, \mathcal{D} \rangle$ of that resource, where:

- $G = \{P, B\}$ is the social relationship graph among those peers P that the peer i considers as relevant for its computation of the trust model $\mathcal{T}^i$. B is the friend (buddy) relationships between peers in P, if available.
- $\Omega = \{\rho(i, e) \mid i \in P, e \in E_{i,j}, j \in P\}$ is the set of all ratings among peers the set P, which can include all ratings in peer serving or rating behaviors.
- F is a rating aggregation function that operates on G, Ω, and outputs estimated values of the function δ_q for each quality attribute q of the resource given every defined context $\pi(q)$. That is, the function F estimates the expected-to-deliver quality level $\delta_q : \Pi_{x \in \pi(q)} D_x \rightarrow D_q$ for every $q \in Q_s$.

The above trust learning algorithm estimates the personalized belief of the peer i on the provider peer j in providing its resource s with a certain quality level. Indeed, it is the formal description of several well-known trust evaluation algorithms in the literature, depending on how we define the social graph G and the aggregating algorithm F. Specifically, let $P_j^0 = \{x \in A \mid E_{x,j} \neq \emptyset\}$ and $P_j^k = \{x \in A \mid l \in P_j^{k-1}, E_{x,l} \neq \emptyset\}, k \geq 1$. Then P_j^0 actually denotes the set of all peers ever do some transactions with j and P_j^k encompasses all peers whose transactions may be considered as relevant to the computation of j's trustworthiness. Most reputation systems, for example (Xiong & Liu, 2004) and other weighted average approaches, only consider $P = P_j^0$ as the set of peers relevant for the reputation computation. Some other approaches that are based on social network analysis define $P = P_j^{k+1} \cap \{x \in A \mid E_{i,x} \neq \emptyset\}$ with higher values of k, e.g, Yu and Singh (Yu *et al.*, 2004). The relationships B between peers in P, which can be learned from the transaction history amog peers, are also used for the reputation estimation (Ashri *et al.*, 2005). Some global trust metrics like EigenTrust (Kamvar *et al.*, 2003) and complaint-based algorithm (Aberer & Despotovic, 2001) consider all peers P in the networks and consider the recommendation/ratings among each pair of them.

3.2.5.5 Peer and Behavior Modeling

The serving behavior of a peer i for a specific request to one of its provided resource $s \in S_i$ at a time $t \in T$ is a random variable sa taking values in its serving action space $\mathcal{SA} = \{gs, bs, is\}$. Each of individual serving actions gs, bs, is respectively corresponds to the action of the peer to provide a resource with high effort (good behavior), with low effort (bad behavior), or simple ignore the request. The distribution of such serving behavior at a time $t \in T$ is modeled as probability distribution $DistS(i, s_i, t)$.

Similarly, the rating behavior of peer i on a transaction with another peer j providing a resource s_j at time t is a random variable ra taking values in a rating action space $\mathcal{RA} = \{hr, ar, br, ir\}$. Each rating action hr, ar, br, ir means that the peer provides honest, advertising, badmouthing ratings or simply doesn't leave any rating after its transaction. The distribution of such rating behavior over time is $DistR(i, s_j, t)$ for a time $t \in T$.

The arrival time and up time of peers in the system can be modeled as two probability distributions $DistArr$ and $DistUp$, respectively. To model the distribution of resources provided by peers and the requests by peers for certain resources, we need to come up with a content distribution $DistC(t)$ and associated request distribution $DistReq$. These distributions can be observed from real networks, for example from current observations in file sharing P2P systems (Klemm *et al.*, 2004; Stutzbach & Rejaie, 2006) and applied in the simulation correspondingly. The simulation framework also enables plugging of different distributions for each of those models flexibly.

3.2.5.6 Decision Making Algorithms

Let $US_i : S \times \mathcal{SA} \rightarrow \mathbb{R}$ be the payoff function of a peer serving a resource. That is, $US_i(s, sa)$ is the gain of the provider i when providing a resource s with a certain behavior sa. The function US_i depends much on the price of the resource and the provider's cost when offereing it with a certain level of efforts. In generally, with every peer i and for every resource s, $US_i(s, bs) \geq US_i(s, gs) \geq US_i(s, is) = 0$.

Similarly, we denote $UU_i : S \times \mathcal{SA} \rightarrow \mathbb{R}$ the payoff function of a peer using a resource. Thus $UU_i(s, sa)$ is the gain of the resource user i when using a resource s with a certain behavior sa, depending how the peer values the resource. For every peer i and every resource s, $UU_i(s, bs) \geq US_i(s, is) = 0 \leq UU_i(s, gs)$. The cost of a rating on a transaction e is $cr(e) > 0$.

Given the above payoff functions, cost and price of resources, the other domain knowledge about the operation of the system, a strategic (rational) peer need to make the fundamental decisions:

- $\mathcal{DS}$: estimate the level of effort (a serving action $a \in \mathcal{SA}$) a peer should exert given a request from a consumer.
- $\mathcal{DR}$: estimate ehavior should a rating peer exhibit (a rating action $a \in \mathcal{RA}$) after a transaction with a provider.
- $\mathcal{S}el$: select a provider $i \in A$ among the set of providers offering similar resources.

3.2.6 Implementation

A full-fledged implementation of the presented simulation framework is developed using the Repast library (North *et al.*, 2006). The underlying simulated network layer is implemented as realistic churn behavior and King latency model. This simulated network can be easily overridden to implement a specific communication network with different topologies, churn and latency models with little efforts.

At the PeerApplication layer, many abstract classes implementing basic functionalities of a peer, associate learner and decision makers are provided to facilitate the simulation of different applications. The system is highly flexible, e.g., distributions on various input such as arrival time, uptime, resource/content and peer behaviors can be easily created by overridden the base/abstract classes.

I have implemented several trust learning algorithms in the TrustManagement subsystem: complaint-based algorithm (Aberer & Despotovic, 2001), EM (Vu & Aberer, 2007) (a generalization of MLE (Despotovic & Aberer, 2005)), OnlyLastRating (Dellarocas, 2003), PeerTrust (Xiong & Liu, 2004). Other baseline algorithms like weighted average algorithms and ideal algorithms for easy comparison of their effectiveness are also provided and more algorithms are being added. Some decision making algorithms to support strategic peers in choosing the best strategy of reporting and serving are also developed.

The following features are supported in the current framework, a more detailed overview of which is available online[1]:

- Support online visualization and automatic collection of simulation results;
- Highly flexible simulation environments with many standard and/or extensible input parameters and performance metrics;
- Ability to dynamically control the simulation environment via GUI and during runtime;
- Simulation can be done interactively or in batch mode with (nested) range of input parameters (thanks to RePast);
- Easy and flexible plugging of customized trust learning and decision making algorithms for empirical evaluation of their performance.

Fig. 3.14 shows some screenshots of our prototyping and simulation framework in action. Further details can be found online. Some more advanced features are being planned, such as the modeling and simulation of dynamic attack scenarios.

3.2.7 Using the Simulation Platform in this Thesis and Beyond

The simulation framework developed is used extensively in the research work done in this thesis. The two concrete use cases of this simulation framework are the two works presented in Chapter 4 and Section 5.2. In each case I use the framework to simulate various complex interaction scenarios, which are intractable analytically.

First, the simulation framework is used in a later work (Chapter 4) to study whether cooperation emerges under many different conditions: (1) peers may join and leave dynamically with realistic churn models; (2) there are different types of peers with different reporting and serving behaviors: including strategic, malicious, and honest; and (3) peers may use different reputation-based trust learning algorithms to estimate the potential trustworthiness of their partners before committing in a transaction.

The simulator is then reused in Section 5.2 to simulate a secret sharing scenario. Peers are delegates keeping shares of a secret key and our goal there is to select the most reliable peers in the system to minimize the number of keys lost. There, peer trustworthiness is defined by a different trust model, depending on their social relationships with the others.

[1] http://lsirpeople.epfl.ch/lhvu/download/repsim/

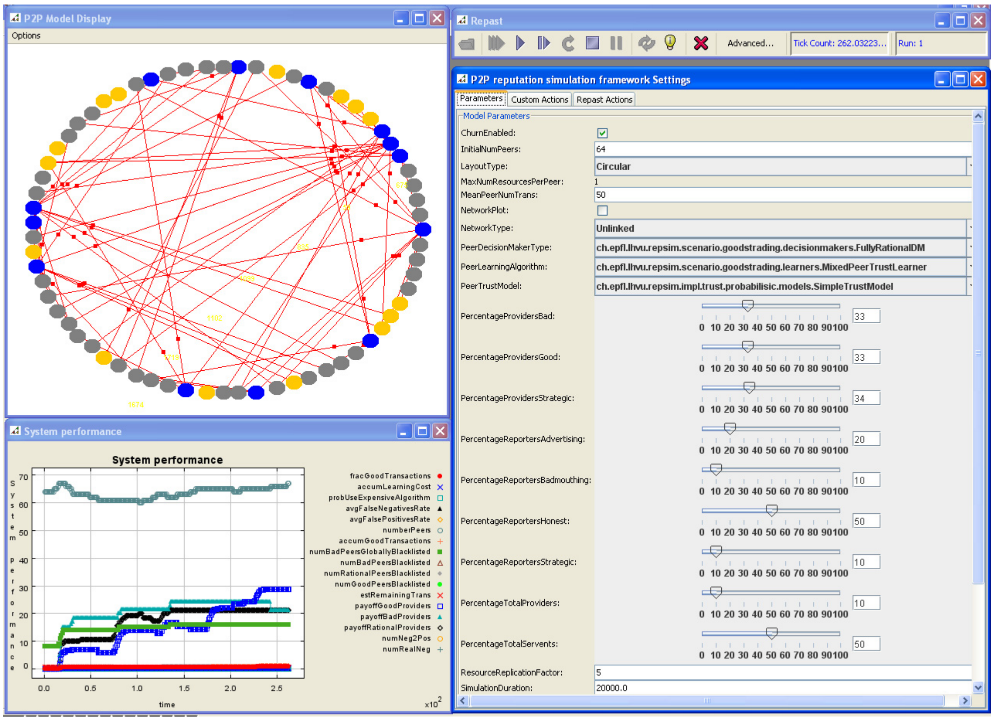
There are many further applications of the simulation platform that are subject to future work and beyond the scope of this thesis. For example, it is straightforward to use the simulator to study the competition of peers with diverse behavioral models to see the effectiveness of a trust learning algorithm and emergence of trust under (1) limited availability of information (2) in presence of communities and trust.

It is also interesting to study in dynamic environments with few interactions and limited information available, which strategy is better for rational peers: simple probabilistic learning algorithms or complex strategic (reinforcement learning) ones.

Several extensions can be made to the framework. The first important extension is to model common, serious attacks on a reputation-based trust system to help testing the robustness of new trust model against them. With extensive collection of such attack models under different simulation settings, the building of a trust benchmark, e.g., similar to the TREC benchmark in information retrieval community, is possible.

The simulation manager can be implemented with a distributed simulation engine to support larger-scale, more realistic simulation of real applications. Even more, emulation of real applications to measure performance of a trust model is also possible. This can be done by implementing an attractive interactive game in which both real people and intelligent agents using newly developed trust models can participate and compete with each other. As the system is totally under control of the simulation manager, from such an emulation the researcher may easily collect reliable and realistic data on behavior of people and ground truths of transaction outcomes among these users. Such data is an invaluable resource for the research community in understanding rationality of users in complex environments.

The current implementation of the simulation tool is flexible, e.g., various simulation inputs can be configured and input during run-time. At the same time the tool can be easily extended to support more features, e.g., new trust learning algorithms can be plugged-in relatively effortless. Scalability is an issue the current implementation has not yet resolved: it is possible to simulate up to roughly 2000 of peers with small trust models, and thus improvements to that regard are desirable.



Tick Count: 262.03223...
Run: 1

Model Parameters

ChurnEnabled: ☑
InitialNumPeers: 64
LayoutType: Circular
MaxNumResourcesPerPeer: 1
MeanPeerNumTrans: 50
NetworkPlot: □
NetworkType: Unlinked
PeerDecisionMakerType: ch.epfl.lhvu.repsim.scenario.goodstrading.decisionmakers.FullyRationalDM
PeerLearningAlgorithm: ch.epfl.lhvu.repsim.scenario.goodstrading.learners.MixedPeerTrustLearner
PeerTrustModel: ch.epfl.lhvu.repsim.impl.trust.probabilisic.models.SimpleTrustModel
PercentageProvidersBad: 33
PercentageProvidersGood: 33
PercentageProvidersStrategic: 34
PercentageReportersAdvertising: 20
PercentageReportersBadmouthing: 10
PercentageReportersHonest: 50
PercentageReportersStrategic: 10
PercentageTotalProviders: 10
PercentageTotalServents: 50
ResourceReplicationFactor: 5
SimulationDuration: 20000.0

Figure 3.14: Screenshots of the trust prototyping and simulation framework in GUI (interactive) mode. The system can be used to simulate interactions and cooperation among peers in different application scenarios by implementing a peer to provide appropriate services. Users can configure various settings of the environments (right panel) such as fraction of peers with different behaviors, computational trust models to be used. The trust relationships among peers are shown in the network of the top-left panel. E.g., in the screenshot honest participants are blue peers with many trust relationships. One can inspect and change behavior of certain peers during simulation time interactively. System performance over time is shown in the bottom-left panel. Most simulation settings such as distribution of different peer types with various behaviors and definitions of (new) system performance metrics can be done programmatically via provided APIs.

Chapter 4

Effective Usage of Computational Trust Model in Heterogeneous Environments

Computational reputation-based trust models using statistical learning have been intensively studied for distributed systems where peers behave maliciously. However practical applications of such models in environments with both malicious and rational behaviors are still very little understood. As the main performance criteria of computational trust models are their statistical accuracy in detecting malicious behaviors, it is my interest in this chapter to study the relation between their accuracy measures and their ability to enforce cooperation among participants and discourage selfish behaviors. I provide theoretical results that show the conditions under which cooperation emerges when using computational trust models with given estimation accuracy and how cooperation can be still sustained while reducing the cost and accuracy of those models.

Specifically, I propose a peer selection protocol that uses a computational trust model as a dishonesty detector to filter out unfair ratings. I prove that such a model with reasonable false positives and false negatives in estimating rating reliability can effectively build trust and cooperation in the system, considering rationality of participants. These results reveal two interesting observations. First, the key to the success of a reputation system in a rational environment is not a sophisticated trust learning mechanism, but an effective identity management scheme to prevent whitewashing behaviors. Second, given an appropriate identity management mechanism, a reputation-based trust model with moderate accuracy bound can be used to enforce cooperation effectively in systems with both rational and malicious participants. In heterogeneous environments where peers use different algorithms to detect misbehavior of potential partners, cooperation still emerges as long as these algorithms have a moderate misclassification error bound. I verify and extend these theoretical results to a variety of settings involving honest, malicious and strategic players through extensive simulation. These results will enable a much more targeted, cost-effective and realistic design for decentralized trust management systems, such as needed for peer-to-peer, electronic commerce or community systems.

More extensively, I prove that even if cheap pseudonyms are possible, there exist certain pricing mechanisms to foster cooperation among rationally opportunistic providers when offering their services in most of their transactions. Subject to certain application-dependent conditions on the temporary gain by cheating of each provider, the mechanism yields small inefficiencies and less risks for participants.

The work in this chapter is published in (Vu & Aberer, 2008a,b, 200x).

4.1 Introduction

Reputation information has long been shown as an effective tool to enforce cooperation and build trust among participants in a variety of e-commerce systems and online forums such as eBay[1], Yahoo Auction[2], or recommender systems[3]. Due to the open nature of such applications, participants usually have incentives not to cooperate, e.g., not to ship the items after receiving payments in an auction site, or not to contribute any resource to others, i.e. free-riding in content sharing systems. In those cases reputation based on user feedback can be an effective measure to isolate bad participants and promote good behaviors.

A reputation mechanism itself relies on truthful feedbacks from participants to be successful. Therefore, it is also vulnerable to many types of attacks, most notably resulting from malicious and rationally opportunistic behaviors. *Malicious users* (malicious peers) are those wanting to take the system down at any cost with strategic attacks. For instance, malicious peers may submit biased ratings to confuse the system and to make it less accurate in learning peers' behaviors. *Rational* (and sometime *opportunistic*) peers are intelligent agents who are neither completely dishonest nor fully cooperative but opportunistic and adapt their behaviors strategically to maximize their expected life-time utilities.

Existing reputation system designs can be generally classified into two classes depending on which of these two behaviors they counter. The first solution class includes *computational trust models* that predict peers' behavior, e.g., whether and to which extent a peer offers high quality services and gives reliable recommendation on the others, based on their historical performance. Such methods are designed to be resilient against different types of strategic attacks from malicious users. The second solution category comprises *game-theoretical approaches* that use reputation as a sanctioning tool to enforce cooperation and establish trust among *rational* participants, e.g., by penalizing bad peers and rewarding good ones. The system is designed as a game whose structure ensures that being cooperative is the strategy in equilibriums of all rational peers who want to maximize their life-time expected utilities.

Computational trust models propose statistical or ad-hoc heuristic methods to aggregate ratings on past transactions of a peer and related participants, from which to compute a trust metric as indication of a target's trustworthiness. Work in this class typically exploits the

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

[2] http://auctions.yahoo.com/

[3] http://www.amazon.com/

signaling role of reputation: peers learn participants' behavior from their reputation as a form of collective social knowledge. The main principle of such models is to predict the future behaviors of peers by examining their past behavioral patterns. Peer behaviors are assumed to follow certain probabilistic models, that is, peers behaving well in the past are likely to continue doing so in forthcoming transactions. For example, a peer may estimate the reliability of another as proportional to the similarity between its own experience with ratings reported by the latter (Xiong & Liu, 2004). The EigenTrust model (Kamvar *et al.*, 2003) evaluates the trustworthiness of a peer according to a global trust metric similar to a PageRank-like value applied on a graph with nodes as peers and edges as peers' recommendations to each other. Alternatively, a learning peer may assume behaviors of a target to follow a certain probabilistic distribution. The trustworthiness of the target is then revised as the posterior probability it provides good services, using ratings by others as evidences to update the original probabilities (Patel *et al.*, 2005; Whitby *et al.*, 2005).

On one hand, computational trust models have been intensively studied and demonstrated to be robust under various (strategic) attacks by malicious peers, e.g., ballot-stuffing, bad-mouthing, or collusion among participants. In fact, statistical learning models and appropriate heuristics can effectively filter out biased information to obtain a correct picture of the peer's historical quality, as empirical evidences have shown (Anceaume & Ravoaja, 2006; Despotovic, 2005; Sun *et al.*, 2006). On the other hand, such computational trust models strongly assume the probabilistic nature of all participants, ignoring the fact that many of them have economic incentives and behave rationally.

In the presence of *rational* peers, who adapt their behaviors strategically to maximize their expected life-time utilities, it is still unclear how well a computational trust model can enforce cooperation in the system. As a typical example, strategic peers can first cooperate to build reputation and then start cheating to increase their life-time utilities. Although some simulations (Liang & Shi, 2008; Schlosser *et al.*, 2005) suggest that these algorithms may also enforce cooperation in presence of both rational and malicious behaviors, *no theoretical analysis has been performed to justify this.*

Game theoretical approaches to trust management, the second solution class, mainly deal with rationally opportunistic participants, who have full knowledge of the solution being used and behave strategically to maximize their life-time utilities. Promising solutions include giving monetary incentives to peers to motivate their good service provisioning and truthfully reporting behavior (Jurca & Faltings, 2006; Miller *et al.*, 2005). Reputation is mostly used as a sanctioning tool: peers with lower reputation are less likely to be selected (Dellarocas, 2005a). However, these solutions usually rely on many assumptions, such as peers being fully rational and having unlimited computational power to find complex equilibriums of the designed mechanism. More importantly, *such solutions do not consider malicious behaviors* of attackers.

Since both rationally opportunistic and malicious behaviors can be present in real environments, an effective reputation management approach to minimize the influence of these unwanted behaviors is of paramount importance. A natural question is how we can exploit well-tested and accurate computational trust models to effectively establish trust among partners in environments where peers may exhibit various behaviors, including honest, malicious, and rationally opportunistic. More concretely, it is important to know whether and under which conditions computational trust models that are resilient against malicious attackers can also motivate cooperation of rationally opportunistic players.

The works of (Friedman & Resnick, 2001) and (Douceur, 2002) give us an initial result: it is a must to have an effective identity management mechanism with high-entrant cost for any participant. Otherwise any reputation model would be vulnerable to Sybil attacks and whitewashing behavior due to cheap identities. However, it is still unknown that given a system with an effective identity management scheme, *whether and under which sufficient conditions a reputation-based computational trust model can enforce trust effectively, given the presence of both malicious and rationally opportunistic behaviors.* As the main performance criteria of such models are their statistical accuracy in detecting malicious behavior, it is our main interest in this chapter to study the relation between such statistical accuracy measures and their ability to enforce cooperation among peers and discourage selfish behaviors.

Another question is related to the optimized usage of computational trust models, as such mechanisms can be costly to deploy and maintain. To effectively learn of bad behaviors and minimize their influence, various costs are incurred, namely the cost to retrieve recommendations, to filter out biased information, to evaluate and update reputation scores of peers (Liang & Shi, 2008). In fact, given the rationality of peers, it may be unnecessary to always use accurate yet costly learning algorithms to encourage truthful behaviors. The reason is rationally selfish peers make use of their knowledge about the deployed algorithm(s) to avoid having bad reputation and maintain their high benefits from the system. Being aware of the existence of such learning algorithms that can reliably detect bad behaviors, peers have little incentives to cheat and thus expensive learning can be avoided. Consequently, accurate yet costly trust learning mechanisms may be needed as an inspection tool to detect past cheating behaviors, in order to punish bad agents appropriately, e.g., by not selecting them for later transactions. Such punishment meted out to peers found cheating can be used to provide sufficient incentives for cooperation of peers. Exploiting this fact, we want *to minimize the cost of using expensive computational trust models* in environments where most peers are rational. A solution to this question implies many benefits for various applications, e.g., where peers have limited resources and quick decisions are generally preferred to avoid missing opportunities, especially in competitive scenarios.

This chapter proposes and analyzes a simple but effective way of using a computational trust model to minimize the influence of malicious peers and at the same time, keep rational peers being cooperative during most of their transactions. Specifically, I propose *a peer selection protocol* that consists of two steps. First, a computational trust model is used to estimate the reliability of the most recent rating on a peer, based on the rating's credibility of the rater and the trustworthiness of the peer being evaluated. It is interesting that such consideration of the most recent rating gives sufficient incentives for rational peers to cooperate most of their transactions. Well-experimented computational trust models, e.g., (Patel *et al.*, 2005; Xiong & Liu, 2004) can be used to identify malicious rating behaviors with certain accuracy in this step.

The result of learning is then used to decide whether to include the target peer for selection or to ignore/blacklist the peer being rated. It is noteworthy that this work is only specific in its assumption of the selection protocol. Though better protocols are feasible, the current one allows us to study the dependencies of the accuracy of learning and cooperation in the system in a general way. Thus, *results obtained in this work are generally applicable* in case of any computational trust mechanisms being used and in presence of many realistic user behaviors. Also, *many existing reputation-based trust approaches are in fact covered by the proposed selection protocol*. Thanks to its simplicity, this protocol can be applied easily in various open e-markets or online communities with different degree of centralization. Theoretical analysis and extensive simulation under various scenarios provide the following results:

- *sufficient conditions on statistical accuracy of a computational trust model to enforce cooperation*: I prove that if statistical accuracy measures of the chosen computational learning algorithm in the first step are good enough, rational peers find it most beneficial to cooperate in all but some of their last transactions. These conditions also show that in rational environments, even simple and naive algorithms may lead to a good social outcome with high cooperation level in the system. According to extensive simulations, such results are still applicable if peers join and leave dynamically, provided most of them staying long enough. The key to enforcing cooperation in such environments would be the existence of an identity management scheme to effectively prevent whitewashing behaviors, rather than the trust learning algorithm being used.
- *a cost-efficient trust management approach*: given cost/accuracy trade-off among different computational trust learning algorithms, I propose a way to combine them to maximize cooperation and build trust in the system, while minimizing cost. Inspired by an inspection game-theoretic approach (Avenhaus *et al.*, 2002), I prove that for the evaluation of rating reliability, under certain assumptions it is sufficient to use an accurate (and expensive) computational trust model with only a low probability while still maintaining high cooperation in the system. As a result, the total implementation cost of the whole selection protocol can be reduced significantly.
- *an identity premium-based pricing mechanism to enforce cooperation*: I prove that even if cheap pseudonyms are possible, with the use of an appropriate pricing scheme that allows peers staying longer in the system to sell services and products at higher prices, the above peer-selection protocol also ensures that rational peers find it optimal to cooperate and provide good services in most but the last of their transactions. Under certain assumptions on the size of a temporary gain by cheating, such a pricing mechanism also yields little inefficiencies to the participants and less risky for the peers who buy the products or services. This result is important, as it offers a countermeasure to whitewashing behaviors given the possibilities of cheap identities in several online business applications.

To the best of my knowledge, this is the first work studying the relation between estimation accuracy of a trust learning model and its ability in enforcing cooperation in open environments where peers exhibit both malicious and rational behaviors. As a theoretical contribution, this work bridges the gap between (Douceur, 2002; Friedman & Resnick, 2001) and existing works studying properties of various computational trust models. I show that beside the necessity of an effective identity management to avoid whitewashing behavior, in rational environments *bounded error rate of the computational trust models being used is sufficient*. Therefore, in rational environments, researchers and practitioners may want to focus more on developing better identity management scheme rather than on more complicated computational models to detect misbehavior: the accuracy requirement of the misbehavior detection model only comes second. As a practical contribution, the methods presented in this chapter can be applied to ensure full cooperation of rational sellers either by (1) by explicitly imposing a sufficiently high entrant cost for newcomers, or (2) by implementing an identity premium-based pricing scheme in which a user with longer interaction history is given more opportunities, e.g., is introduced to more buyers, thus implicitly increasing the value of his identities overtime, which demotivates the user from cheating, leaves and rejoins under new identities (whitewashing).

This analysis also implies that *any existing trust learning algorithm in the literature, if shown to have a moderate error bound, shall achieve similar effect*: formulating the social optimum point in the system where rational participants mostly cooperate. In practice, this means cooperation is possible in heterogeneous environments, where *peers may use different learning algorithms to learn trustworthiness of their potential partners.* Cooperation still emerges as long as these algorithms guarantee such an accuracy bound. The work most related to this one is by Dellarocas (Dellarocas, 2005a) that studies some design parameters of a reputation system and its effectiveness. However, his work does not include the analysis of the influence of malicious behaviors and the optimization of the cost of the reputation mechanism being deployed. The work of this chapter analyzes the impact of these factors, comprises extensive simulations on a variety of scenarios, and is also applicable for any open system with different degrees of centralization. In a larger context, the presented peer selection protocol establishes an umbrella framework to use reputation information effectively in decentralized and self-organized systems by exploiting both its *signaling* and *sanctioning* roles (Dellarocas, 2005a).

Section 4.2 of this chapter presents a typical example scenario and then gives a formal definition of the system model. In Section 4.3 I study the relation between accuracy of a computational trust model and its incentive-compatibility with a detailed theoretical analysis. I then propose a way to combine different trust models to minimize their usage cost while retaining their efficiency in enforcing cooperation among rational participants in Section 4.4. Section 4.5 proposes an approximate approach to use computational trust models in scenarios with peers joining and leaving dynamically overtime. The simulation and experimental results are presented in Section 4.6. I summarize related work in Section 4.8 before giving some discussions and concluding the chapter in Section 4.9.

4.2 System Model

4.2.1 Example applications and assumptions

Throughout the chapter, the following running example is used to illustrate different concepts and results of this work. We consider a decentralized (peer-to-peer) market of products or services, where each participant can be provider and/or client of certain services. As a concrete example, any person in the Internet can advertise and sell their services in a peer-to-peer system and/or an online social network like Facebook[1] or MySpace[2]. Another realistic showcase of this is the recent launch of Neighborhoods[3] that enables eBay users to do shopping via their social networks. Henceforth, we use the notions of a client, a provider, or a service to illustrate our concepts in a market of services. In other application contexts such as in auction sites, these notions may also mean a seller, a buyer, and resources/products/articles advertised by a seller respectively.

Given the above application scenario, clients can search and buy available services that match their needs and interests. As a common rule, a client has to pay for the service first. Only after receiving the payment, the provider may provide the service with either low or high quality, e.g., in eBay this means either the seller cheats by not shipping the articles or the articles are not of good quality as described. The traditional way to enforce the cooperation of the provider via reputation mechanism is to allow a client to rate a provider as *honest* or *cheating* after finishing a transaction with him (Resnick *et al.*, 2000). Other clients can search for available recommendations or ratings on the provider and decide whether they should go into transaction with this provider. In this case, a computational mechanism may is very helpful in effectively eliminate unreliable ratings and minimize influences of malicious rating users. Other incentive mechanism to elicit sufficient truthful reports from clients for such computational mechanism or trust learning algorithm are also necessary.

Whereas the problem of trust and cooperation is ubiquitous and may be present cross all layers of the system, in this work we would focus on the cooperation among participants in the application layer. That is, we assume the existence of a (possibly decentralized) storage system that supports efficient storage and search for publicly available information, namely information on resources and related ratings. Such storage system must ensure that advertisements of provided services, published ratings, and transaction information are authentic and can not be tampered with. That is, peers can neither fake transactions nor modify feedback data submitted by others. Although addressing these security goals are beyond the scope of our paper, I believe existing solutions are available. For example, these goals can be met by using a DHT-based storage (Aberer *et al.*, 2003a) and cryptographic tools such as digital signatures based on a decentralized PKI infrastructure (Datta *et al.*, 2003). Also, centralized storage solutions such as provided by *eBay* can be considered sufficiently secure.

We would also assume that peers know the lower bound of prices of a service (or a service or a

[1] http://www.facebook.com
[2] http://www.myspace.com
[3] http://neighborhoods.ebay.com

resource) u_* and they can estimate the gain v of a cheating provider peer after each transaction. For example, v approximates the service value as evaluated by the provider plus the shipping cost. Such an assumption is realistic: a centralized system can define those values u_* for each category of services or products, or peers can learn these values by looking at trading history of other peers in the system. Note that v may be less than the service value u, e.g., the client still receives the service, which has worse quality than as that of the original description.

The final assumptions are that peers stay in the system long-enough so that reputation information has any effects. This assumption can be relaxed under certain circumstances, e.g., in centralized environments where it is possible to impose an entrant fee for a newly joined peer. In practice, this assumption also holds since identities are not cheap: users must invest time and money to build relationships with others before they can participate in business transactions with others. Therefore, it is better for most users to stay as longer in the system rather than departing and start everything all over with a new identity. Such an extension of the analysis will be considered later on (Section 4.7).

The above simplifying assumptions *are standard and well-accepted in the trust and reputation research communities* (Despotovic, 2005). More importantly, the system model is rather realistic: it is in fact an abstraction of many practical P2P application scenarios with different degrees of centralization and where participants are rational to a certain extent. Such a scenario can represent, for example, a centralized eBay-like trading site, a social-network based system for selling and buying goods, or a decentralized markets of computational or storage services (Buyya *et al.*, 2001; Papazoglou & Georgakopoulos, 2003). Consequently, the proposed solution can be used in all these applications.

4.2.2 System model

The above applications can be formulated as a P2P application where each participant plays the role of a provider or a client of certain services, namely a sellable article or a computational or storage service, with certain prices and quality. Denote P the set of all peers and $W : P \times P \rightarrow D_f$ the social relationship among them, where D_f is the domain to represent the relationship values between two peers. For example, D_f may represent the relationship type between two friends in a social network (family/close/normal/stranger) or the weight of an edge between two nodes in a trust network[1], whichever is applicable.

Denote as $\mathcal{O}$ the set of outcomes of a transaction between peers. In this work, we consider $\mathcal{O} = \{0, 1\}$ as a binary set corresponding to a bad or a good outcome. A peer provides a service with price u for its clients, where u lies within a range of minimal price u_* and maximal price u^*. Thus we name u the "legitimate" payoff (or gain) of a provider peer in a transaction if it behaves honestly, e.g., providing a service with high quality. The transaction in this case is considered as having a good outcome to the client[2]. In the opposite case if the provider cheats,

[1] http://trust.mindswap.org/

[2] A user's rating on a transaction may be different from the real transaction outcome because of two reasons: on purpose if the user is not honest, or accidentally if there is observation noise. For simplicity, we do not consider the noise in this analysis. However, if there is a (small) probability that a provider with honest behavior still

namely it provides a bad service, the provider gains a further “illegitimate” amount v, where $0 \leq v \leq v^* < \infty$ and the transaction outcome is considered as bad. The upper bound v^* typically depends on the maximal price u^* of a service, and is only necessary in our theoretical analysis. In practice rational peers need to know only v for each transaction they involve in.

Define Ω the set of all transactions in the system and $\Omega(x, y)$ the set of all transactions where the peer x is a client and the peer y is the provider. For convenience, let $S(x)$ be the set of peers having provided services to x. Let $r(x, y, tr) \in \mathcal{O} = \{0, 1\}$ be a binary rating from a peer x on another peer y on a transaction $tr \in \Omega(y)$.

4.2.3 Computational trust models as dishonesty detectors

A peer (the learning peer) may use a computational trust model to learn behavior of another (the target) via various statistical or heuristic methods and based on various information sources. The first source comes from the performance statistics of the target in past transactions. These statistics are possibly collected from recommendations/ratings by previous partners and from personal experience of the learning peer on the target. Other information includes intrinsic features of the target itself, e.g., frequencies of posted ratings and involved transactions, location of the raters (Cornelli *et al.*, 2002), relationships with other peers in the systems (Ashri *et al.*, 2005). The behavior that can be learned from such information include both service and rating behavior of a peer: the former represents the likelihood that the peer offers a high quality services to its clients. The latter means the trustworthiness of the peer in rating a transaction, that is, whether a client truthfully reports its experience.

In this work, a computational trust model is used as a dishonesty detector to effectively evaluate the trustworthiness (reliability) of a rating, similar to a conventional spam filter (Graham, 2002). From this perspective, a trust mechanism can be abstracted as in Definition 4 (see Section 4.2.2 for the meanings of related concepts).

Definition 4 *A (personalized) reputation-based* computational trust model *used by a peer i to estimate the trustworthiness of a rating by another peer j is a 5-tuple $\mathcal{R} = \langle P_i, V_i, \mathcal{F}_j, A, D \rangle$ where:*

- *$P_i \subseteq P$ is a set of peers that i considered relevant to the evaluation of j's rating reliability.*
- *$V_i \subseteq \{r(x, y, tr) \mid x \in P_i \wedge y \in P_i, tr \in \Omega(x, y)\}$ is set of ratings related to peers in P_i.*
- *$\mathcal{F}_j$ is a set of properties of the target peer j, for instance, its location, frequencies of posted ratings, number of involved transactions, etc.*
- *A is a function that operates on $P_i, W, V_i, \mathcal{F}_j$ and outputs a trust value T_{ij}, where W is the social relationships among peers as introduced earlier. The value T_{ij} is understood as the estimated trustworthiness of j in giving the current rating. T_{ij} may take binary, real, or discrete values.*
- *D is some decision rule(s) with binary outcome $\{1, 0\}$, stating whether i should trust the rating by j given the (personalized) trust metric T_{ij}.*

yields a bad transaction outcome to the client, inclusion such a probability in our approach is straightforward.

Definition 4 is an abstraction of several (if not most) well-known heuristic or statistical trust evaluation algorithms in the literature. This formal model includes all approaches studying trust on peer-to-peer systems (Despotovic & Aberer, 2006), on social networks (Golbeck, 2006), in e-commerce systems and other online communities (Dellarocas, 2005b; Jøsang *et al.*, 2007).

The set of relevant peers and related ratings P_i and V_i constitutes the actual feedback retrieval mechanism of the computational trust model and contributes the main cost of the mechanism, as explained later on. Examples 1 and 2 specify two typical trust learning models proposed by (Xiong & Liu, 2004) and (Vu & Aberer, 2007) using the formalism of Definition 4.

Example 1 *The personalized trust model PeerTrust (Xiong & Liu, 2004) can be used to evaluate rating reliability via the following formalism:*

- $P_i = S(i) \cap S(j)$. *In other words, the set of relevant peers* P_i *includes those peers* k *having provided services to both* i, j.
- $V_i = \{r(x, k, tr) \mid x \in \{i, j\}, k \in P_i, tr \in \Omega(i, k) \cup \Omega(j, k)\}$, *that is,* V_i *consists of those ratings by* i *and* j *on service behaviors of other peers* k *in the relevant peer set* P_i.
- *the relationship* W *among peers and the features of the target* $\mathcal{F}_j$ *are not considered.*
- $T_{ij} = 1 - \sqrt{\frac{\sum_{k \in P_i} \frac{\sum_{t_{ik} \in \Omega(i,k)} r(i,k,t_{ik})}{\|\Omega(i,k)\|} - \frac{\sum_{t_{jk} \in \Omega(j,k)} r(j,k,t_{jk})}{\|\Omega(j,k)\|}}{\|P_i\|}}$ *be a measure of similarity between possible ratings* $r_s(i, k, t_{ik})$ *and* $r_s(j, k, t_{ik})$ *on those transactions* $t_{ik} \in \Omega(i, k)$ *and* $t_{jk} \in \Omega(j, k)$.
- *A rating by* j *is considered as reliable if* $T_{ij} > T_{min}$ *and as unreliable otherwise, where* T_{min} *is a (possibly global) system design threshold. Alternatively, we can normalize* T_{ij} *into* $[0, 1]$ *and trust the rating with probability* T_{ij}. *With probability* $1 - T_{ij}$, *the rating is distrusted (evaluated as unreliable).*

Example 2 *Another approach to estimate peer's trustworthiness is to assume peers to behave according to a probabilistic model: similarity in rating on one target leads to similarity in rating on another target (Vu & Aberer, 2007). From the viewpoint of the learning peer* i, *the target peer* j *has a probability* T_{ij} *of reporting truthfully what its observes. This model is specified similar to Example 1, except the definition of* T_{ij} *and the decision rule* D_i *to estimate rating reliability. We formally define:*

$$T_{ij} = \frac{\sum_{k \in P_i, t_{ik} \in \Omega(i,k), t_{jk} \in \Omega(j,k)} I(r(i, k, t_{ik}) = r(j, k, t_{jk}))}{\|P_i\| \sum_{k \in P_i} \|\Omega(i, k)\| \sum_{k \in P_i} \|\Omega(j, k)\|} \tag{4.1}$$

where the indicator function $I(c)$ *evaluates to* 1 *if the boolean condition* c *is true. Thus* T_{ij} *is defined by the fraction of ratings by* j *having the same values as ratings by* i. *This* T_{ij} *is an estimate of the probability the peer* j *being honest when rating, and such an estimate maximizes the likelihood of having the observation set* V_i *by the set of relevant peers* P_i. *The decision rule* D_i *is usually probabilistic, e.g., trust the rating with probability* T_{ij} *and distrust it with probability* $1 - T_{ij}$.

Another simple yet important computational trust model is considered in Example 3.

Example 3 *The naive computational trust model is defined as* $\mathcal{N} = \langle \emptyset, \emptyset, \emptyset, \mathcal{I}, D_{\mathcal{I}} \rangle$*, where algorithm* $A = \mathcal{I}$ *always outputs* $T_{ij} = 1$ *and the set of decision rules* $D_{\mathcal{I}}$ *stating that a rating by* j *is always considered as reliable. In other words, the naive model* $\mathcal{N}$ *simply trusts all ratings by any peer.*

Other computational trust models can be represented similarly. Global trust metrics like EigenTrust (Kamvar *et al.*, 2003) and complaint-based algorithm (Aberer & Despotovic, 2001) consider all peers in the networks, thus $P_i = P$, and consider the recommendation/ratings among each pair of them.

Considering a computational trust model as a dishonesty detector, we define its statistical accuracy measures similar to those of a conventional spam filter (Graham, 2002) in Definition 5.

Definition 5 *The (estimation)* accuracy of a computational trust model $\mathcal{R}$ *formulated in Definition 4 in estimating the reliability of a rating is defined by its two* misclassification errors, $0 \leq \alpha, \beta \leq 1$. *Specifically,* $\alpha =$*Pr(rating estimated as reliable by* $\mathcal{R}$ $\mid$ *rating is actually unreliable), and* $\beta =$ *Pr(rating estimated as unreliable by* $\mathcal{R}$ $\mid$ *rating is actually reliable). We also refer to* α *and* β *respectively as* false positive *and* false negative *rate henceforth.*

The accuracy of a computational model also implies its resilience to possible malicious attacks that manipulate ratings, by estimating the trustworthiness of a rating using the combination of performance statistics of both the rating peer and the provider peer to estimate the trustworthiness of a rating. Note that the actual value of α, β of a dishonesty detector may change (possibly improved) over time, depending on the learning capability of the computational trust model being used.

4.2.4 Strategic peer-selection protocol

We propose the following approach for a a rational peer (client) to select a provider among candidates (Definition 6). The concrete implementation of this mechanism in a general dynamic setting will be presented later on in Section 4.5.

We note that the goal of this protocol is to enable us study the relation between the accuracy measures of a computational trust model being used and its abilities to enforce cooperation in a general way. In fact better selection protocols may be available.

Definition 6 *A peer (a potential client) uses the following* peer-selection protocol $\mathcal{S}_k = \langle \mathcal{R}, k \rangle$ *to evaluate the eligibility of a provider for its transactions:*

1. *the client gets the most recent binary rating* r *on the provider, considering the absence of a rating as the presence of a positive one.*

2. *the binary reliability* $\widehat{t}$ *of* r *is evaluated with the computational trust model* $\mathcal{R}$*.*

3. *if* $\widehat{t} = 1 \wedge r = 0$ *or* $\widehat{t} = 0 \wedge r = 1$*, the client publishes this information (detect of a reliable negative rating) to a shared space (a global black list).*

4. *the provider is included for selection if in the shared space there are less than $k \geq 1$ published cheating detections about the provider. Otherwise the client ignores the provider.*

Please refer to Table 4.1 for the most frequently used notations. Fig. 4.1 shows the sequence diagram of the above peer-selection protocol, which relies on two hidden assumptions. Firstly, at each steps, it is assumed that there are a sufficient number of potential clients (at least k) for each provider. This is realistic since otherwise there is very little incentive to peers to participate in the system anyway. Secondly, a globally blacklisted provider can not do any further transaction with his current identity. The second assumption limits the current analysis to the case identities are costly to obtain for peers and will be relaxed later on in this chapter.

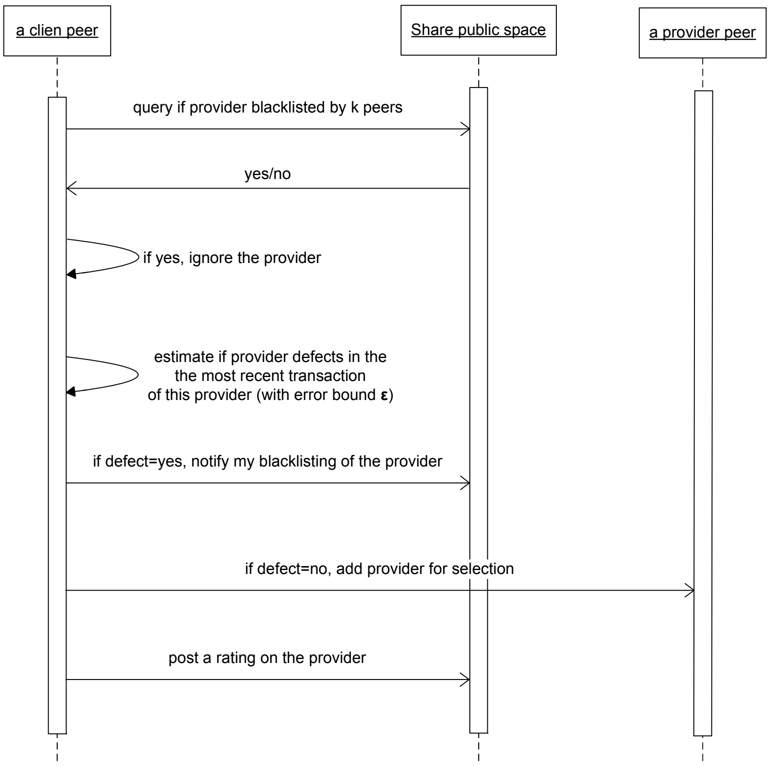

a clien peer
Share public space
a provider peer
query if provider blacklisted by k peers
yes/no
if yes, ignore the provider
estimate if provider defects in the the most recent transaction of this provider (with error bound **ε**)
if defect=yes, notify my blacklisting of the provider
if defect=no, add provider for selection
post a rating on the provider

Figure 4.1: Sequence diagram for the peer-selection protocol in Def. 6.

The above peer-selection protocol $\mathcal{S}_k$ is tough for bad players, including malicious and rationally opportunistic providers. It assures that a globally blacklisted provider has no further chance to provide services and gains revenues from the system. The evaluation of rating reliability by a computational trust model in step (2) aims to reduce influences of strategic manipulation of ratings by rational or malicious peers. The goal here is to eliminate as much malicious providers as possible when they start cheating and incentivize rationally opportunistic providers to cooperate. The use of the above peer-selection protocol with a global computational trust model mimics the behavior of a centralized reputation system in practice. The parameter $k \geq 1$

represents the cautiousness of a peer in trusting cheating detections published by others. We do not directly consider the incentives to leave any feedback after a transaction in detail in this work. Nevertheless, the absence of a rating after a transaction as a positive one s.t. in case of few ratings, appropriate decisions can still be made. In fact, it is possible to integrate existing incentive mechanisms, e.g., via side-payment (Miller *et al.*, 2005) into our system without major effort. Furthermore, clients have other indirect incentives to leave reliable ratings after transactions. First, such behaviors help to eliminate bad providers. Second, providers are motivated to cooperate with honestly reporting clients as shown in a later analysis (Corollary 2), which also give additional motivation for peers to leave truthful feedback.

Similarly to the above issue, providing incentives to share data for the learning step (evaluation of the rating reliability) is an orthogonal issue to the current analysis. In fact, such incentives for sharing the learning results are not much of an issue: in case all clients do not share the learning results, a client can still do the learning by itself and our approach still holds if the evaluation of rating reliability is verifiable so that writing wrong learning results can be easily detected. Generally such an assumption is realistic, e.g., the learning at step (2) is verifiable in case of global computational trust models such as EigenTrust (Kamvar *et al.*, 2003) or complaint-based (Aberer & Despotovic, 2001). Otherwise, it is possible to design the system so that peers also post data related to their learning on the defection of a provider in the most recent transaction to the shared public space, or provides such data on-demand for the querying peer(s).

Thus the main problem is the badmouthing attack, when many peers collude to badmouth a certain provider to eliminate it. To reduce the effect of this attack, a robust algorithm $\mathcal{R}$ should consider the trustworthiness of both the rater and the provider being rated to estimates whether a rating is reliable, to reduce the undesired impact of observation noise. The accidental blacklisting of a good peer is of no harm to a client and as we will analyzed later on, can be reduced by both increasing k and lowering the error bound of the second step by using a more expensive and sophisticated trust model.

4.3 Accuracy of Computational Trust Models as Their Capabilities to Foster Cooperation

In this section, we study the relation between the accuracy of a computational trust model and its capabilities to enforce cooperation. That is, we analyze the possibility to effectively use a computational model with reasonably good accuracy for boosting trust and cooperation in decentralized environments.

4.3.1 Quantifying the accuracy of a computational trust model

Misclassification errors of a given computational model depend on several factors, one is the design of the computational model itself. Other important factors are personal experience of the learning peer, feedback retrieval aggregation strategies, and underlying probabilistic model.

Table 4.1: Commonly used notations

Notation	Definition
u_*	minimal price of offered services/resources/services, $u_* > 0$
u^*	maximal price of offered services/resources/services $u^* \geq u_*$
u	legitimate gain of provider when cooperating, $u_* \leq u \leq u^*$
v	additional gain of provider when cheating in a transaction, $v > 0$
v^*	maximal additional gain of provider by cheating, $0 \leq v_* < \infty$
$\mathcal{R}$	a computational trust model to estimate reliability of a rating, explained in detail in Definition 4
$\mathcal{N}$	the naïve computational model in Example 3
α	Pr(rating estimated as reliable | rating is actually unreliable)
β	Pr(rating estimated as unreliable | rating is actually reliable)
k	# of posted cheating detection on a provider before it is globally ignored by other clients
$\mathcal{S}_k = \langle \mathcal{R}, k \rangle$	a peer-selection protocol specified in Definition 6
ε	upper-bound of α and β, $0 \leq \varepsilon \leq 1$
Δ	# of remaining transactions for which a rational provider does not have incentives to cooperate, $\Delta \in \mathbb{N}$

Also influencing the accuracy are certain exogenous factors such as behaviors of participants (malicious, rationally opportunistic, or voluntarily cooperative) and system dynamics (such as the volumes of transactions and participation levels over time).

We present the following results on quantifying accuracy measures of a computational trust model used by a rational peer to estimate the reliability of a rating. These results can be seen as guidelines to design and select suitable computational trust models as appropriate dishonesty detectors in this work.

Proposition 3 *Given a computational trust model $\mathcal{R} = \langle P_i, V_i, \mathcal{F}_j, A, D \rangle$ as in Definition 4, where $\mathcal{R}$ is publicly known to every peer. Suppose that raters are rational and want to maximize the misclassification errors of the detection. If D yields a deterministic, publicly known outcome at each evaluation step, in equilibrium $\alpha = \beta = 0.5$. Thus the use of the model $\mathcal{R}$ as a dishonesty detector in this scenario yields no better result than random guessing.*

Proof 3 *Let t and $\widehat{t}$ be the binary reliability of a rating, as exhibited by the rater and as estimated by the learning peer, respectively. Denote $\mathcal{H} = \langle P_i, W, V_i, \mathcal{F}_j \rangle$ the input of the algorithm A (c.f. Definition 4), we have:*

$$\alpha = Pr(\widehat{t} = 1 \mid t = 0, \mathcal{H}) \propto Pr(\widehat{t} = 1, t = 0 \mid \mathcal{H}) \tag{4.2}$$

If the model $\mathcal{R}$ is publicly known and the decision rules D are deterministic, a rational rater knows exactly whether the rating is estimated as reliable given the history $\mathcal{H}$. Therefore, the rater can strategically provide rating with an opposite reliability to maximize misclassification *errors of the dishonesty detector. The game between a learning peer and the rater is shown in Fig. 4.2.*

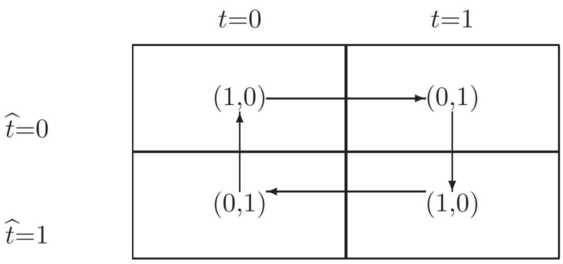


Figure 4.2: The game between a learning peer (the row player) and the strategic rater (the column player). The notation (x, y) means that the payoff of the learning peer is x and the rater is y. Since the goal of a rational rater is to maximize misclassification errors of the dishonesty detector, we give a rater a payoff 1 if $t \neq \widehat{t}$ and 0 if $t = \widehat{t}$. The payoff of a learning peer is the opposite.

The arrows in Fig. 4.2 denote the possible moves of each player to maximize its payoff, showing that the game has no pure Nash equilibrium. The only mixed equilibrium of the game is: the rater exhibits a random rating strategy $Pr(t = 1) = Pr(t = 0) = 0.5$ and the learning peer estimate the rating reliability as $Pr(\widehat{t} = 1) = Pr(\widehat{t} = 0) = 0.5$, so $\alpha = \beta = 0.5$.

Example 4 quantifies misclassification errors of the naive dishonesty detector introduced in previous section.

Example 4 *Consider the simple computational model $\mathcal{N}$ that uses a naively optimistic algorithm: trust all ratings and consider the absence of a rating as a positive one. We claim that misclassification errors of $\mathcal{N}$ are $\alpha = \alpha_0 = 1$ and $\beta = \beta_0 = 0$. In fact, let $0 \leq \delta_h, \delta_l, \delta_i \leq 1$, where $\delta_h + \delta_l + \delta_i = 1$ be respectively the probabilities that the rating peer provides a reliable rating, an unreliable one, and no rating after a transaction with a specific provider. Denote $est+$ (resp. $est-$) the events that the most recent rating is evaluated by the learning peer as reliable (resp. unreliable), and let $real+$ (resp. $real-$) be the events that the rating is actually reliable (resp. unreliable). There are two possibilities:*

- *the provider did cooperate in the last transaction, thus the absence of a rating is equivalent to the presence of a reliable positive rating. So $\alpha_0 = Pr(est+, real-)/Pr(real-) = \delta_l/\delta_l = 1$ and $\beta_0 = Pr(est-, real+)/Pr(real+) = 0/(\delta_h + \delta_i) = 0$.*
- *the provider did not cooperate in last transaction, thus the absence of a rating implies the presence an unreliable positive rating. Still, we have $\alpha_0 = Pr(est+, real-)/Pr(real-) = (\delta_l + \delta_i)/(\delta_l + \delta_i) = 1$ and $\beta_0 = Pr(est-, real+)/Pr(real+) = 0/\delta_h = 0$.*

Actually, misclassification errors α, β of a computational trust model may change (possibly improve) over time, depending on the learning capability of the trust computation algorithm.

In this work, we generally do not care about actual values of those errors α, β but only their upper-bound ε, and we are concerned with those computational models with accuracy better than random guess, i.e., $\varepsilon < 0.5$. Although Proposition 3 show cases where such bounds can not be achieved, practically accurate learning with α, β upper-bounded by some $\varepsilon < 0.5$ is feasible in most application scenarios. Such scenarios include those where details of algorithm A, e.g., the rating history to be used for the trust estimation, or the decision rule D are private information of the learning peer, and there are various types of raters, even with deterministic and irrational behaviors. It is noteworthy that the learning peer still can prove the correctness of its estimation $\widehat{t}$ of the reliability of a rating (as a clarification of results posted in Step 2 of the selection protocol in Def. 6). This proof can be done simply by revealing relevant information it has used in the trust model $\mathcal{R}$ to the querying peer. In those cases misclassification errors of the associated computational trust model as a dishonesty detector can be measured by empirical experiments. This question has already been extensively studied in several previous work, namely (Kamvar *et al.*, 2003; Levien, 2002; Xiong & Liu, 2004), most of which having been shown to have low α, β under various attack scenarios. Several measurements on α, β of the computational trust models of Examples 1 and 2 have also been performed under different conditions, which confirmed that achieving an accuracy bound $\varepsilon < 0.5$ is possible when estimating the reliability of ratings (c.f. Section 4.6.3).

Another example of an accurate yet expensive method to estimate reliability of ratings by peer on a provider is to perform full monitoring on performance of the provider to learn its real *past* behavior. Such monitoring can be implemented in many ways: in an e-commerce system, monitoring is possibly done via legal investigations on suspicious transactions conducted. In a market of Web services, one can deploy monitoring agents to periodically probe and test the service being offered by a provider to estimate the real offered quality level offered by that provider during a specific period.

4.3.2 Using computational trust models with certain accuracy for cooperation enforcement

Recall from Section 4.2 that providers have a minimal legitimate gain u_* and a maximal illegitimate cheating gain v^*, and their possible strategies at each transaction are either to cooperate or cheat. Given a bound ε for α, β of the computational trust model(s) being used by clients in the system, Theorem 1 shows the relation between the error bound ε of a computational trust model and its effectiveness in enforcing cooperation of a provider during its life-time.

Theorem 1 *Given the peer-selection protocol $\mathcal{S}_k = \langle \mathcal{R}, k \rangle$ where the computational trust model $\mathcal{R}$ has misclassification errors α, β upper-bounded by $\varepsilon < 0.5$. Suppose that identities are difficult or very costly to obtain for any participants. If we have in addition:*

$$\varepsilon < \varepsilon_{max}(k) = 1/(1 + \sqrt[k]{1 + v^*/u_*}) \tag{4.3}$$

Define

$$\Delta = max\{1, \lceil \frac{\ln{[1-\frac{v^*\varepsilon^k}{u_*((1-\varepsilon)^k-\varepsilon^k)}]}}{\ln{(1-\varepsilon^k)}} \rceil\} \tag{4.4}$$

1. *it is optimal for a rational provider to cooperate in all but its last Δ transactions*[1].

2. *it is optimal for a rational provider to cooperate in the current transaction whose illegitimate gain is v, if it stays for more than Δ_v further transactions, where Δ_v is computed as in Eq (4.4) with v in place of v^*.*

3. *let N_h be the number of transactions where an a fully cooperative (honest) provider can participate till it is mistakenly blacklisted, and let N_c be the number of bad transactions a malicious provider can benefit from the system until it is globally eliminated from the system respectively. We have $E[N_h] > 1/\varepsilon^k$ and $E[N_c] < 1/(1-\varepsilon)^k$.*

The above results hold even in presence of strategic manipulation of ratings by providers.

Proof 4 *We first prove (1). Rational providers apparently do not find incentives to cooperate in the last transaction. Consider those rational providers staying in the system for $\Delta > 1$ more transactions after the current one.*

Let $0 \leq h, s, l, i \leq 1$ respectively be the probabilities that the current client exhibits following rating behaviors after the transaction: honest (provides reliable ratings), advertising (posts positive ratings on the provider), badmouthing (rates the provider negatively), and non-participating (does not leave any rating), where $h + s + l + i = 1$. Note that possible strategic rating manipulations by any raters colluding with the current provider are all considered by the above probabilities. For example, consider the case where the provider may use a fake identity to stuff a positive rating with a newer timestamp to hide its cheating in a transaction. In this case, the provider still has additional gain v in the transaction, and the dishonesty detection is applied on the fake rating whose rater is a client with completely advertising behavior, i.e., $h = i = l = 0, s = 1$.

The probabilities that an honest provider obtains a positive (resp. negative) rating after a transaction are $h^+ = h + s + i = 1 - l$ (resp. $1 - h^+$). The honest provider is blacklisted if either the true positive rating is not accepted by the computational trust model as reliable (with probability β), or the wrong negative rating is accepted as reliable (with probability α). Thus the probability that the provider will be blacklisted by a forthcoming client is: $x_b = h^+\beta + (1-h^+)\alpha = (1-l)\beta + l\alpha \leq \varepsilon$, since $0 \leq l \leq 1$ and $0 \leq \alpha \leq \varepsilon, 0 \leq \beta \leq \varepsilon$.

The probability the provider is globally blacklisted after the current transaction is then $x_b^k \leq \varepsilon^k$. This inequality holds even in presence of malicious or strategic manipulation of ratings by any raters with different h, l, s, i, provided that misclassification errors α, β of $\mathfrak{R}$ are less than ε.

Similarly reasoning, if the provider is cheating in this transaction, the probability it obtains a positive rating is $l^+ = s + i = 1 - h - l$. With probability $1 - l^+$ such a provider receives a

[1] With small ε^k, we have $\Delta = max\{1, \lceil v^*/(u_*((1-\varepsilon)^k - \varepsilon^k))\rceil\}$, as in previous work (Vu & Aberer, 2008a)

negative rating. In this case, the provider will be blacklisted by a future client with probability $y_b = l^+(1-\alpha) + (1-l^+)(1-\beta) = (1-h-l)(1-\alpha) + (h+l)(1-\beta) \geq 1-\varepsilon$. *Thus the probability the provider is globally blacklisted is* $y_b^k \geq (1-\varepsilon)^k$.

Let U *be the current (accumulative) utilities of a rational provider and* u_h *be its best (maximized) expected utilities for the remaining time in the system if it is not globally blacklisted after the current transaction. Denote* U_{honest} *(and* U_{cheat}*) as the best (maximal) expected life-time utilities of the provider if it is honest (respectively cheating) in the current transaction, it follows that:*

$$\begin{aligned} U_{honest} &= U + u + u_h(1 - x_b^k) \\ U_{cheat} &= U + (u + v) + u_h(1 - y_b^k) \\ \delta_{hc} &= U_{honest} - U_{cheat} = -v + u_h(y_b^k - x_b^k) \\ &\geq -v^* + u_h((1-\varepsilon)^k - \varepsilon^k) \end{aligned}$$

One can verify that the above reasoning is applicable in the following two situations: first, identities are very difficult to obtain and thus the provider can not rejoin under a new identity. Second, the cost of obtaining a new identity outweighs the (maximal) temporary benefit gained by cheating in a transaction. We will reconsider the case of cheap identities later on.

As an honest user is still blacklisted with probability $x_b^k < \varepsilon^k$, *one can verify that the fully cooperative strategy of a provider during* $\Delta \geq 1$ *transactions leads to a total utility of at least* $\frac{1-(1-x_b^k)^\Delta}{x_b^k} u_* \geq \frac{1-(1-\varepsilon^k)^\Delta}{\varepsilon^k} u_* > 0$. *Note that for small* x_b^k, $u_h \geq \Delta u_*$ *approximately.*

It follows that $\delta_{hc} \geq 0$ *iff* $\Delta \geq \frac{\ln[1-\frac{v^* \varepsilon^k}{u_*((1-\varepsilon)^k - \varepsilon^k)}]}{\ln(1-\varepsilon^k)}$, *where* $\varepsilon < \varepsilon_{max}(k) = 1/(1+\sqrt[k]{1+v^*/u_*}) < 0.5$ *so that the logarithm is well-defined.*

Therefore, a rational provider considers cooperation as the dominant strategy in any transaction except in its last $\Delta = max\{1, \lceil \frac{\ln[1-\frac{v^* \varepsilon^k}{u_*((1-\varepsilon)^k - \varepsilon^k)}]}{\ln(1-\varepsilon^k)} \rceil\}$ *ones. Thus (1) is proved. The proof of (2) is then straightforward from the above analysis.*

To prove (3), note that after each transaction, the probability that by accident, an honest provider is globally blacklisted is $x_b^k \leq \varepsilon^k$. *In the worst case ever,* N_h *is a geometric random variable with probability* ε^k, *hence* $E[N_h] > 1/\varepsilon^k$.

By similar reasoning, the probability a malicious provider is globally blacklisted is $y_b^k \geq (1-\varepsilon)^k$, *and thus* $E[N_c] < 1/(1-\varepsilon)^k$

Theorem 1 also holds for the case peers use different computational trust models with different inputs and personalized settings to evaluate the trustworthiness of the last rater, and the probability of detecting a bad rater is different for each peer. Furthermore, it is possible to take into account noise in observations. In fact there are (small) probabilities that an intrinsically honest provider appears as cheating to a client, e.g., a good seller may be rated negatively if the article is lost when being shipped to the buyer, and a cheating provider satisfies the client, e.g., a seller sends a low-quality item yet still pleases the client. The inclusion of such probabilities in the above analysis is straightforward.

According to Theorem 1, if the temporary gain v^* is very high, such as the case of selling of expensive items, the parameter $\Delta \to \infty$. This means that enforcing the cooperation of a rational provider in such a transaction is impossible, which is intuitive. In other cases, a relation between Δ and the error upper-bound ε can be drawn. For example, Fig. 4.3 shows the relation between Δ and ε for $v^* = u_*$ and different k, e.g., peers sell and buy items of comparable prices. We have the following observations. First, the required accuracy upper-bound $\varepsilon_{max}(k)$ reaches to 0.5 with larger k values, yet the incentive of cooperations decreases rapidly (Δ becomes very large). Even so, for rational long-term providers who stay in the system infinitely, the number of the last Δ transactions plays no role and thus any trust model with reasonably good accuracy $\varepsilon < \varepsilon_{max}(k)$ can be used as an effective sanctioning tool to motivate providers' cooperation. Given an approximate value of ε, one should select the threshold k appropriately such that $\varepsilon < \varepsilon_{max}(k)$. For a given $\varepsilon < \varepsilon_{max}(k)$, smaller threshold k is preferred.

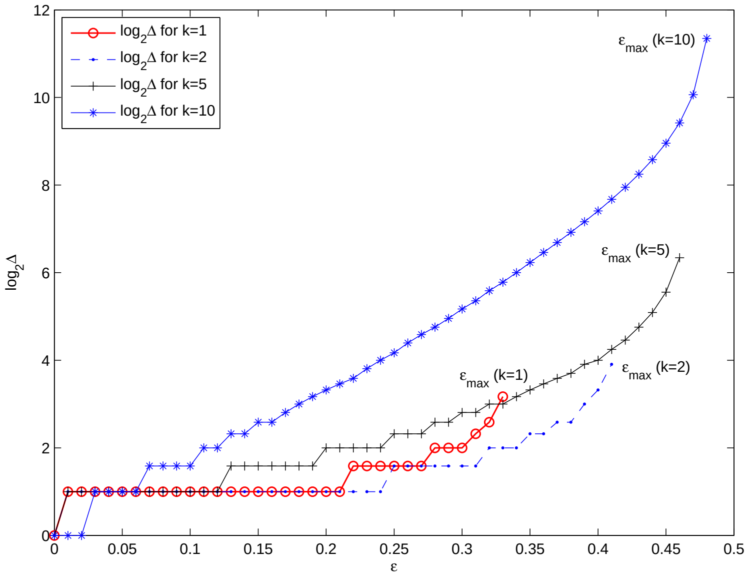


Figure 4.3: The relation between the upper-bound ε of misclassification errors of a computational trust model and incentives of rational providers to cooperate for different values of k where $u_* = v^*$. Rational providers find it most beneficial to cooperate during all but their last Δ transactions.

In certain application contexts where providers only participate in a limited number of transactions, or in case of high k values, very high levels of accuracy ($\varepsilon < 0.05$) are required to reduce the parameter Δ, i.e., to ensure cooperation of providers in most transactions. Consequently, the study of the selection protocol $\mathcal{S}_k = \langle \mathcal{R}, k \rangle$ gives us the following observation on those sufficient conditions to use a computational trust model $\mathcal{R}$ to effectively boost cooperation among rational entities (Corollary 1).

Corollary 1 *It is possible to use any computational trust model with misclassification errors upper-bounded by some $\varepsilon < \varepsilon_{max} < 0.5$ to effectively enforce cooperation of rational providers who participate infinitely or in a very large number of transactions, even in presence of strategic rating manipulation by participants.*

The possibility of eliminating malicious providers and wrongly blacklisting honest ones is given in Fig. 4.4. It is observed that the system behaves much better with lower ε, as intuitively expected. The higher the accuracy of the computational trust model being used (lower ε), the lower the probability an honest provider is accidental blacklisted (higher $E[N_h]$) and the higher the probability malicious providers are eliminated from the system (lower $E[N_c]$). Higher thresholds k reduce the possibilities of wrongly blacklisting honest providers yet also increase the incentives of malicious behaviors. Hence, given a known ε, it is recommended to choose the value of k appropriately depending on the prior information on the environment vulnerability. In environments with more malicious behaviors, it is better for rational clients to choose smaller k values to eliminate bad providers quickly, at the cost of ignoring good providers. In less vulnerable environments where most providers are likely good, higher k is recommended. Note that the given trends in Fig. 4.4 are for the worst case scenario in an extremely vulnerable environment, where an honest provider is repeatedly badmouthed by other users, and a malicious provider has enough resources for disguising her cheating activities by posting many positive ratings to the system consecutively.

The analysis in this section applies in the assumption that there exists an effective identity management scheme that ensures blacklisted providers have to pay a very high cost to enter the system again, and hence *identities are not cheap*. This is realistic in many practical application, e.g., in our running scenario of trading on a social network such as Facebook, users have to spend time and money to build relationships with other users, and it is non-optimal for them to simply cheat, discard their long-time investment on current identities to simply start all over again.

The analysis in this section also provides us a starting point to design a mechanism to ensure full cooperation of rational providers, either by (1) by explicitly imposing a sufficiently high entrant cost $c \geq \Delta v^*$ for newcomers, where Δ is defined from the system parameters ε, k, or (2) by letting a provider with longer interaction history more valuables, e.g., allows the provider to be introduced to more clients, thus increases the value of his long-time identities, which unmotivates him from cheating and escape). This extension regarding (2) will be presented later in Section 4.7.

4.3.3 Capability of the naive trust model in fostering cooperation

We want to further consider one very simple case of the naive trust model $\mathcal{N}$ (Example 3). Corollary 2 shows the relation between the capability of such naively optimistic algorithm $\mathcal{N}$ in enforcing cooperation and the client's truthfully reporting probability h, in case there is

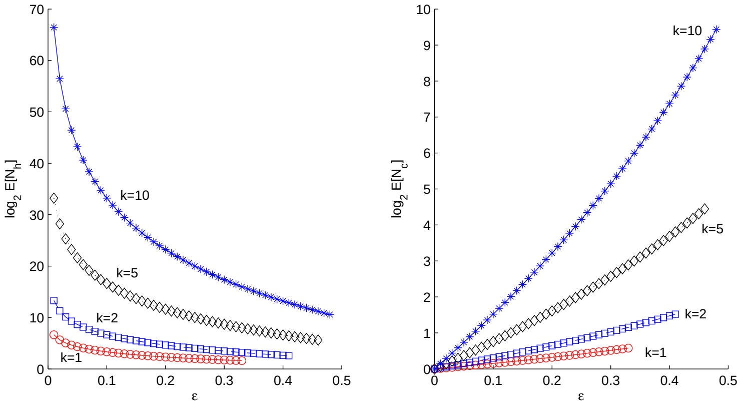


Figure 4.4: The relation between the upper-bound ε of misclassification errors of a computational trust model and a *worst case lower-bound* of $E[N_h]$ (left side). On the right hand side is the relation between ε and the *worst case upper-bound* of the $E[N_c]$. Plots are drawn with $u_* = v^*$ and different k values.

no strategic manipulation of ratings in the system. The peer-selection protocol $\langle \mathcal{N}, 1\rangle$ for this special case is actually equivalent to the reputation system considering only the last rating studied by Dellarocas (Dellarocas, 2005a)

Corollary 2 *Suppose that $1 \geq h > h_{min} = (1 + v^*/u_*)/(1 + 2v^*/u_*)$ is the average probability that a client leaves an honest rating after a transaction. With the exception of strategic rating manipulation by the provider, the peer-selection protocol $\langle \mathcal{N}, 1\rangle$ makes it optimal for a rational provider to cooperate in all transactions but its last $\Delta = max\{1, \lceil \frac{\ln[1-(1-h)/(2h-1)]}{\ln h} \rceil\}$ ones. Such a selection mechanism also gives direct incentives for long-term clients to leave truthful ratings after their transactions.*

Proof 5 *The naive computational model $\mathcal{N}$ has misclassification errors $\alpha = 1$ and $\beta = 0$ (Example 4). Proceed as in the analysis of Theorem 1, we have $\delta_{hc} \geq -v^* + u_h(h - (1-h)) = -v^* + u_h(2h-1) \geq -v^* + (1 - h^{\Delta})(2h-1)/(1-h)$ (herein $u_h \geq (1-h^{\Delta})/(1-h)$).*

Thus a rational provider would cooperate with a client if $\delta_{hc} \geq 0$, or $\Delta \geq \frac{\ln[1-(1-h)/(2h-1)]}{\ln h}$. The condition $1 > h > h_{min} = (1 + v^*/u_*)/(1 + 2v^*/u_*)$ is make sure the logarithm is well-defined.

Note that $\delta_{hc} \geq -v^* + (1-h^{\Delta})(2h-1)/(1-h)$, where the right hand side is monotonically increasing of h. This fact gives direct incentives for a long-staying client to leave a correct rating after a transaction so as to increase the overall probability of reporting truthfully h of any client as estimated by subsequent providers. This maximizes the chance of this current client to have successful transactions in the future even with other rational providers (for larger h, δ_{hc} gets larger and thus it is more favored for the future provider to cooperate than to cheat).

The assumption of no strategic rating manipulation in Corollary 2 implies that users follow *probabilistic behavior*, e.g., whether they leave a good or bad rating depends on their (erroneous) evaluation of a provider's behavior, e.g., due to possible shipping delays or loss. Thus one can estimate overall truthfulness reporting probability h of users in the system. In the presence of (strategic) manipulation of the reports, a client must use a sophisticated trust learning algorithm to evaluate the reliability of the rater, as presented in the previous Section 4.3.2. Otherwise a selection protocol based on only the latest rating can be attacked easily: after cheating in a transaction the provider can collude with another client to immediately stuff a positive rating with newer timestamp in the system.

Fig. 4.5 shows the relation between the overall probability h that a peer is a honest reporting peer and the number of Δ last transactions for which providers may not find incentives to cooperate, with different settings of v^* and u_*. Those cases where $v^*/u_* < 1$ can be explained as a provider gives items with bad quality than the description.

For example, let us consider the previous trading scenario where peers sell and buy services of similar prices ($u_* \simeq v^*$). If provider is a long-term player, and the truthfully reporting probability $h > h_{min} \simeq .8$ (a highly noisy environment), even the naive algorithm makes it optimal for a provider to behave honestly for most of his transactions (except the last two).

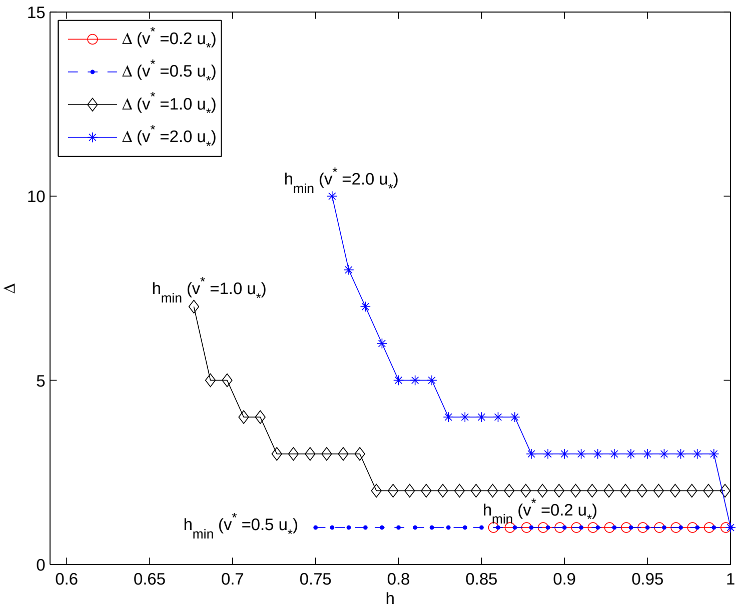


Figure 4.5: The relation between the truthful reporting rate h and the number of last transactions during which providers have no incentive to cooperate in case clients use the selection algorithm $\langle \mathcal{N}, 1\rangle$.

It is important to note that Theorem 1 and Corollary 2 state only those *sufficient*, not *necessary* conditions for a computational trust model to be incentive-compatible. In general, there maybe other designs better than the proposed selection protocol $\mathcal{S}_k$.

4.4 Cost-efficient Reputation Management

4.4.1 Implementation cost of computational trust models

Accurate computational trust models are costly to build and to maintain. Several sources may contribute to their cost, as summarized in Definition 7.

Definition 7 *The implementation cost of a computational trust model $\langle P_i, V_i, \mathcal{F}_j, A, D \rangle$, as in Definition 4 consists of the following component cost:*

- *communication cost τ_c to explore the peers P_i and retrieve relevant ratings V_i. Associated with this cost is the storage cost τ_s to store and to maintain rating information and historical performance of potential partners/raters.*
- *computational cost τ_e of the algorithm A and decision making algorithm $\mathcal{D}$ to aggregate, analyze ratings and make appropriate decisions.*
- *possible monetary cost τ_m: since reliable ratings can be considered as sellable goods offered by third-party monitoring agents in the system, buying them incurs certain cost. This cost may also include the payment for participants in system with any side-payment mechanism to elicit truthful reporting behaviors, such as (Miller et al., 2005).*
- *hidden/opportunity cost τ_o: time-consuming learning algorithms may lead to late decisions and hence more costly than fast algorithms, especially in competitive scenarios.*

A detailed study of the cost model of each computational model is much dependent on the system and application on which we apply the algorithm. For example, the estimation of opportunity cost τ_o during the lifetime of a peer is non-trivial, though it maybe significant. Such a thorough study of all cost types of existing algorithms is out of the scope of this work. Instead we only focus on the most obvious and measurable cost of communication τ_c. Evaluations of such communication cost τ_c of well-known trust learning algorithms will be presented in Section 4.6.

4.4.2 Using a computational trust model cost-effectively

This section studies the cost and benefits of using different trust learning algorithms and investigates the possibilities of minimizing the cost while still maintaining a high level of cooperation in the system. While an algorithm with better accuracy is generally preferable (see Theorem 1), it usually comes with higher cost. For instance, an algorithm relying on the retrieval/buying of reliable information from trusted third-party agents may give us more accurate information about the providers, yet it is apparently more expensive than a simple majority voting algorithm.

Consider the case of a peer that can choose either a computational trust model $\mathcal{R}_1$ or another model $\mathcal{R}_2$ to evaluate the trustworthiness (reliability) of a rating on some provider. Suppose that $\mathcal{R}_1$ is an accurate algorithm with misclassification errors α, β, both upper-bounded by some $\varepsilon < \varepsilon_{max}(k = 1)$, and with an expected cost $\mathcal{C}_1$. A typical example of $\mathcal{R}_1$ is to buy information from some third-party monitoring agents to get correct estimate of the behavior of the last client/rater. Let $\mathcal{R}_2$ be the naive computational trust model $\mathcal{N}$ in Example 3 (always trusts all ratings) with misclassification errors $\alpha_2 = 1, \beta_2 = 0$, and a negligible cost $\mathcal{C}_2 \ll \mathcal{C}_1$. The cost $\mathcal{C}_1$ and $\mathcal{C}_2$ may compose of certain component cost in Definition 7.

On one hand, since α, β are less than $\varepsilon < \varepsilon_{max}(k = 1)$, the computational model $\mathcal{R}_1$ can be used to motivate the cooperation of a provider in most transactions (Theorem 1), yet this approach is costly to deploy. On the other hand, the use of the model $\mathcal{N}$ is much cheaper and more preferable. However, it is impossible to use only the naive model $\mathcal{R}_2 = \mathcal{N}$ for the same purpose since the provider can strategically manipulate ratings easily, e.g., by colluding with others to submit biased ratings.

If we assume that a client uses the computational model $\mathcal{R}_1$ with probability c and model $\mathcal{R}_2$ with probability $1 - c$ when evaluate rating reliability, Theorem 2 proposes a way to optimize the cost of using the expensive computational model $\mathcal{R}_1$ while still ensuring cooperation in the system.

Theorem 2 *Consider the selection protocol $\mathcal{S}_1 = \langle \mathcal{R}, 1 \rangle$, in which the dishonesty detector $\mathcal{R}$ is implemented by using the trust model $\mathcal{R}_1$ with probability c and the naive model $\mathcal{N}$ with probability $1 - c$. Suppose that ε is small, and the incentive for badmouthing is negligible, e.g., where providers provide services of different categories with few competition. The provider finds it optimal to cooperate in all but its last Δ transactions, where $\Delta = max\{1, \lceil \frac{v^*}{u_*(1-2\varepsilon)} \rceil\}$, under the condition that $c \geq c_* = \frac{v^*}{\delta u_*(1-2\varepsilon)}$, where $\delta \geq \Delta$ is the number of remaining transactions of the rational provider. This result holds in presence of strategic manipulations of ratings.*

Proof 6 *Let $\delta \geq \Delta$ be the number of remaining transactions of the provider at the current step. Proceed as in Theorem 1 with $k = 1$, $\alpha' = c\alpha + (1-c)\alpha_2 = 1 - c + c\alpha$, and $\beta' = c\beta + (1-c)\beta_2 = c\beta$, we get $\delta_{hc} \geq -v^* + u_h(h(1-c) + c(1-\alpha-\beta-(\beta-\alpha)h))$. Here, the probability an honest provider is blacklisted is $x_b = (1-l)\beta' + l\alpha' = c(1-l)\beta + l(1-c+c\alpha)$, where l is a small probability someone badmouths the provider. Thus we have $x_b = (1-c)l + c[(1-l)\beta + l\alpha] \leq \max(l, \varepsilon)$, which is small. Therefore, approximately, $u_h \geq \delta u_*$. As a result, $\delta_{hc} \geq -v^* + \delta u_*(h(1-c) + c(1-\alpha-\beta-(\beta-\alpha)h))$.*

Since $0 \leq h \leq 1$, it follows that $\alpha+\beta+(\beta-\alpha)h \leq \alpha+\beta+\max\{\beta-\alpha, 0\} \leq 2\max\{\beta, \alpha\} \leq 2\varepsilon$. Thus, $h(1-c) + c(1-\alpha-\beta-(\beta-\alpha)h) \geq c(1-2\varepsilon)$ for $c \in [0,1]$. This makes $\delta_{hc} \geq -v^* + \delta u_* c(1-2\varepsilon)$. Equivalently, $\delta_{hc} \geq 0$, or being cooperation is a dominant strategy for the provider if and only if $c \geq c_* = \frac{v^*}{\delta u_*(1-2\varepsilon)}$.

According to Theorem 1 (k=1) with small ε, using only the algorithm $\mathcal{R}_1$ can ensure cooperation of a rational provider in all transactions but its last Δ ones. That is $\delta_{hc} \geq -v^* + \Delta u_*(1-2\varepsilon) \geq 0$, or equivalently $\Delta \geq \frac{v^*}{u_*(1-2\varepsilon)}$. Since $\delta \geq \Delta$, one can verify that $c_* \leq 1$ and thus is a valid probability.

Fig. 4.6 shows the minimal probability c_* of using the expensive trust model $\mathcal{R}_1$ depending on the estimated remaining transactions δ of the provider, with $u_* = v^*$ and $\varepsilon = 0.01$. If most providers staying in the system infinitely or long enough, the mixture of two computational trust models help to reduce the total implementation cost significantly, as derived from Fig. 4.6. Hence, given the rationality of participants, the accurate computational trust algorithm $\mathcal{R}_1$ mostly plays the role of a sanctioning tool rather than the role of learning trustworthiness of potential partners.

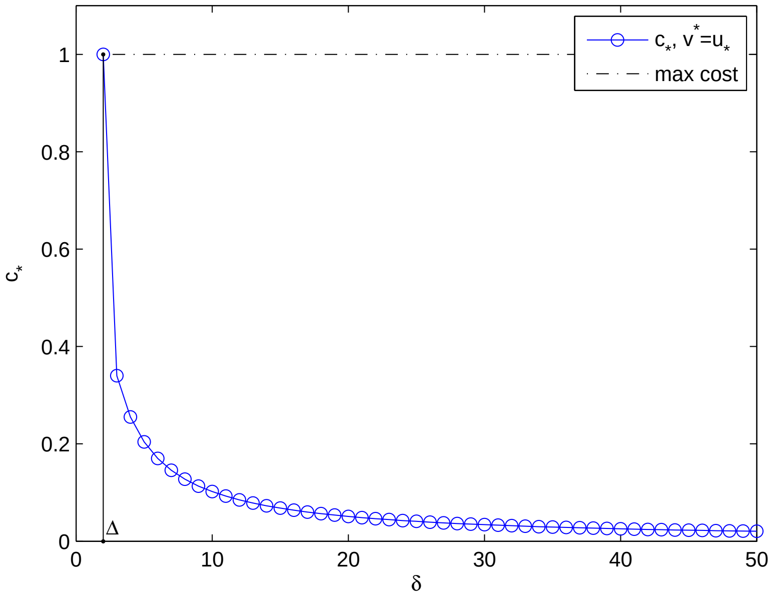


Figure 4.6: The minimal necessary probability c_* to use the expensive algorithm $\mathcal{R}_1$ depending on the number of remaining transactions δ of a rational provider for $\varepsilon = 0.01, u_* = v^*$.

4.5 Using Reputation Information in Dynamic Scenarios

The above analysis shows that if rational peers staying in the system infinitely, cooperation is ensured relatively easily. Based on this result, this section presents a peer-selection protocol to be used by rational peers (as clients) that ensures cooperation of rational providers for most transactions in dynamic scenarios, where peers join and leave dynamically.

Considering peer rationality, solving the problem of cooperation in this scenario is non-trivial, even impossible. Corollary 1 reveals that the key to ensure cooperation is not the accuracy of the computational trust learning algorithms being used but an *identity management mechanism* that is effective in preventing white-washing behaviors. This is in accordance with previous work (Friedman & Resnick, 2001). Thus in this section we must restrict the problem to the case where the *cost of establishing identity is sufficiently high* such that a peer is motivated to use one identity for all its transactions. For centralized/distributed e-trading systems where trusted parties are present, a simple yet effective solution to ensure cooperation of all rational participants even in the last Δ transactions is to require each provider to deposit an approximate amount Δv^* to the third trusted party before being able to join the system. This deposited sum will only be returned to the provider if it intends to quit the system and only if it has not been detected as cheating by k or more others peers. Alternatively, providers can be further introduced to more clients over time, thus making identities with long history more valuable. As a result, rational peers have little incentives to cheat, leave the system, and start all over again with new identities.

Given the above restriction, the following information can be learnt by peers: first, the distribution of number of life-time transactions of providers, which is observable. Similarly, peers can estimate worst case values or upper-bound ε of misclassification errors α, β of the learning algorithm $\mathcal{R}$ being used. In fact, such statistics can be learnt from experience of each participant or available as common knowledge.

Rational peers can use all available information and behave strategically to maximize their long-term benefits. Thus, a rational client has to make decision based on the expected number of remaining transactions of each candidate provider. We approximate the algorithm for a rational (strategic) client to select a provider before each transaction as in Algorithm 3.

Algorithm 3 selectProvider(providers S, alg $\mathcal{R}$, threshold k)

```
Eligibles = $\varnothing$;
Retrieve the global blacklist $L$;
for each $s \in S \setminus L$ do
  Get worst case values $\varepsilon$ of errors $\alpha, \beta$ of algorithm $\mathcal{R}$;
  Estimate benefit $u_*$ and current cheating gain $v$ of $s$;
  Compute $\Delta_{min}$ as in Eq (4.4) with $v$ instead of $v^*$;
  $p[s]$ = Pr[provider $s$ stays at least $\Delta_{min}$ further transactions];
  Get binary rating $r_i$ by peer $i$ on the latest transaction of $s$;
  Run $\mathcal{R}$ to evaluate the reliability (binary trust) $t_i$ of $r_i$.
  if $r_i == t_i$ then
    Eligibles = Eligibles $\cup \{s, p[s]\}$;
  else
    Post the negative detection result $r_i \neq t_i$ on the shared space;
    Put $s$ in the blacklist $L$ if there are at least $k$ such negative results;
  end if
end for
Select provider $s$ from Eligibles with probability $p(s)/\sum_s p(s)$;
```

Algorithm 3 follows the peer-selection protocol $\mathcal{S}_k = \langle \mathcal{R}, k \rangle$ in Definition 6 and computes the involved quantities in a straight forward way. For example, in an e-trading system, the minimal legitimate gain u_* of a provider (a seller) is the minimal value of an article accepted for trading in the system. The gain v is the value of the current item plus the shipping cost announced by the seller. Δ_{min} is the minimal number of remaining transactions that a rational provider finds incentives to cooperate for the worst level of α, β (claim (2) of Theorem 1). Since we assume that the distribution of number of transactions by providers can be learnt from trading history of various providers in the system, it is possible for the client to estimate the probability $p[s]$

that a provider stays in Δ_{min} further transactions. This estimation is personalized to each client and dependent on his own belief on the continuation of the game of the provider with future clients. It is important to note that such an estimation is not known to the provider himself, otherwise the end-game situation would occur: a provider knowing that the client knows of his last transactions would never cooperate. This probability can be seen approximately the probability that this strategic provider cooperates in the current transaction compared and thus is used as a selection criteria (line 17).

According to Theorem 1, the best strategies of rational providers are described as in Algorithm 4. That is, a strategic provider will cooperate in all transaction except its last Δ_* ones. Similar to a client, the provider estimates its Δ_* parameter based on the global knowledge of misclassification errors α, β of the learning algorithm being used, its current temporal gain v and minimal legitimate gain u_* at each step.

Algorithm 4 bestServiceStrategy(alg $\mathcal{R}$, threshold k): servingStrategy

```
Get worst case values $\varepsilon$ of errors $\alpha, \beta$ of algorithm $\mathcal{R}$;
Estimate the minimal own benefit $u_*$ and illegitime gain $v$ in the current transaction;
Compute $\Delta_*$ as in Eq (4.4) with $v$ instead of $v^*$;
Estimate own remaining number of transactions $\delta$;
if $\delta \geq \Delta_*$ then
    Return cooperative;
else
    Return cheating;
end if
```

Using the above peer-selection protocol, the number of successful transactions, or cooperation level, in the system generally depends on how accurate clients can estimate the number of remaining transactions of a provider in order to select the right one still having incentives to cooperate. More concretely, a strategic provider is motivated to cooperate if $\delta \geq \Delta_*$.

The algorithm for a strategic client to use a mix of two algorithms $\mathcal{R}$ and $\mathcal{N}$ to select a provider (Theorem 2) is then implemented as follows. A client estimates the number of remaining transaction δ_{client} of a provider from expected number of transactions of all providers in the system. Next, it computes the minimal probability $c_{client} = \frac{v}{\delta_{client} u_*(1-2\varepsilon)}$ it needs to use $\mathcal{R}$ in the current transaction. On the other hand, the best thing a strategic provider can do is to also estimate those values $c_{client}, \delta_{client}$, and computes the minimal probability $c_{provider} = \frac{v}{\delta u_*(1-2\varepsilon)}$ for its actual remaining number of transactions δ. The provider cooperates only if it stays more than $\delta > \Delta_*$ and $c_{client} \geq c_{provider}$.

Since it is very difficult to perform a rigorous analysis on the peer cooperation in such dynamic scenarios with various possible parameters of the join and leave processes of peers, we defer the measurement and analysis of cooperation level in the system until Section 4.6, where extensive simulation with various input parameters obtained from real case studies is performed for such analysis purposes.

4.6 Experimental Results

I use the generic trust prototyping and simulation framework as presented in Chapter 3 as the simulation tool for all experiments in this section. This tool enables the rapid development and testing of many trust computational models under a variety of environment settings. Particularly, new algorithms for peers and appropriate performance metrics can be easily defined and integrated into the framework without major efforts.

4.6.1 Simulation goals and setting

In practice full rationality is usually impossible to implement, but may be approximated by strategic behaviors. In our simulation, I do such an approximation where peers gather all available information in the system to make the best decision in order to maximize their lifetime utilities. My purpose is to answer the following question: given that the one implementing rational peers has the knowledge of the above theoretical results and implement their rational peers accordingly, whether the cooperation in the system still emerges.

The goal of my simulation is then to investigate whether (1): in approximately rational environments, reasonably good trust learning algorithms may help to enforce cooperation in many transactions, even with small observation noise, and (2): in such an approximate rational setting, costly learning algorithms can be used with a small fraction of time, without much affecting the cooperation.

An example P2P trading application is simulated with the prototyping and simulation framework, presented in Chapter 3. The details on the modeling, implementation, and configuration files for all experiments in this chapter are also available[1].

Peers are modeled as sellers and/or buyers of many articles and exhibit different behaviors. Cooperation level and accumulated utilities of different peer types who sell and buy goods of different prices are then measured and analyzed under various simulations settings: first, peers can leave and join dynamically, thus having different number of transactions during their lifetime. Second, buyers use different computational trust models with different personalized inputs and settings to evaluate the rating trustworthiness of others. Third, peers exhibit several behaviors, both irrationally malicious and/or strategic, details to be presented in coming sections. To more accurately model the delay of information propagation in the network, the routing of messages is implemented so that the search cost in the system similar to that of a logarithmic P2P routing overlay, i.e., it takes time $O(logn)$ to lookup a data item in system with n peers. The latencies among nodes in the network are set up with the King latency data set provided by (Gummadi *et al.*, 2002), which measured the real latencies between nodes in the Internet. The dynamic joins and leaves of peers in the system are simulated according to statistics from real-life case studies on many live peer-to-peer systems (Stutzbach & Rejaie, 2006). Specifically, I schedule the inter-arrival time of peers to follow a Weibull distribution with shape parameter 0.53, and generate the up-time of peers to follow a Weibull distribution with shape parameter

[1] http://lsirpeople.epfl.ch/lhvu/download/repsim/

0.34 (Stutzbach & Rejaie, 2006). The scaling parameters of these Weibull distributions are set so that all peers live long enough to send and receive many messages during their up-time.

Since I do not have statistics of the number of transactions a peer performs in a real peer-to-peer trading scenario, such a distribution is approximated in my experiments as follows: I compute the number of transactions a seller participates from its up-time by using a scaling factor K. K is set such that those peers with up-time approximating the mean of the overall uptime distributions participate in μ_{trans} transactions, where μ_{trans} is a parameter of the simulation. The rationale behinds this choice is that the number of transactions a peer involves is assumed to be proportional to the user's participating time in the system. Experiments with other distributions of the number of peer transactions are subject to future work.

4.6.2 Implementation of peer behaviors

Since it is impossible to implement full rationality, in our simulation rational peers are approximated as strategic ones who use all available information to find the best strategy for them at each step. A (strategic) buyer first searches for the available articles satisfying its requirements and then selects one according to Algorithm 3 and its variant introduced previously in Section 4.5.

A *good seller* ships the article after receiving the payment with a very high probability ($\gamma^+ = 0.99$) of satisfying the buyer and get a good rating. A *bad seller* generally either doesn't ship the article or ship a low quality item to the buyer, resulting in a very low probability ($\gamma^- \approx 0.05$) of meeting buyer's anticipation and thus likely to get a bad rating afterwards. Other sellers are *strategic* and use all available information to find the best strategy to follow, e.g., whether it should ship the item or not, so as to maximize its long-term utilities, following the strategy described in Algorithm 4 (Section 4.5). It is also possible that the observation noise γ^+, γ^- may vary depending on other factors, e.g., subject to the buyer's viewpoint given the price of an item. The inclusion of these details into the analysis and in my simulation is feasible. However, these issues are subject to future work.

Regarding rating behaviors, peers consist of the following types: *honest* (always reports correctly about what it has observed), *badmouthing* (always reports negatively), *advertising* (always reports positively), *ignoring* (does not report), and *strategic*. Strategic raters can provide correct, incorrect ratings or do not leave any rating depending on the situation. To reduce the complexity of the simulation, in this work the rating behavior of strategic peers is limited to the following *safe* strategy. The reporting strategy of a peer A on its own estimate of the trustworthiness of another peer X when asked by a peer B is determined based on the relationships between A and B, and between A and X. Specifically, if the querying peer B is unknown to A, where the target peer X is a trusted peer, a blacklisted, or unknown peer of A, A would respectively report positively, negatively, or honestly. If a trusted peer B asks A about trustworthiness of another peer X, A always reports honestly. Requests from blacklisted peers are ignored. This reporting strategy is chosen for two reasons. First, the cost of reporting after a transaction in our current application is negligible. Second, this strategy helps peers to

Table 4.2: Experimental settings of representative simulation scenarios with different types of sellers and buyers. Seller types consist of: s% strategic, g% good, and b% bad. There are five rater types modeled: h% honest, sr% strategic, a% advertising, b% badmouthing, the rest are those peers leaving no reports after a transaction. $a > b$ since in most case studies of current reputation-enabled systems, majority of collective feedback are positive ratings, thus it is likely that advertising behaviors are popular than badmouthing. In each scenario by default $k = 1$ if it is not specified otherwise.

Scenario	s	g	b	h	sr	a	b	**Result**
C_1. No strategic sellers, most seller bad	0	15	85	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.7
C_2. No strategic sellers, half seller good	0	50	50	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.7
C_3. No strategic sellers, most seller good	0	85	15	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.7
C_4. Few sellers strategic, most bad	10	5	85	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.8
C_4^k. Similar to C_4 with $k = 1, 2, 5$	10	5	85	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.9
C_5. Most sellers strategic	85	5	10	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.8
C_6. Approximately equal seller types	33	34	33	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.8
C_6^k. Same as C_5, thresholds $k = 1, 2, 5$	33	34	33	0 to 100	$\frac{100-h}{5}$	$\frac{2(100-h)}{5}$	$\frac{100-h}{5}$	Fig. 4.9
C_7. All sellers strategic	100	0	0	0 to 100	$\frac{99-h}{3}$	$\frac{(99-h)}{3}$	1	Fig. 4.10

rapidly build up and extend its trust relationships with many other peers in the system, thus *intuitively most beneficial* to them.

Experimental settings and corresponding results are summarized in Table 4.2. The trading system is simulated in most cases with dynamically leaves and joins of peers, starting with $n = 256$ peers. In the stationary regime the number of peers approximately double this initial number with our churn model. The simulator is able to run with up to 1024 initial peers, and the results are similar. Each simulation is run until stationary regime and the measures of each metric are their medians over *at least* 35 different runs (to avoid outliers).

4.6.3 Accuracy of example computational trust models

Three computational trust models are implemented and used in the simulation as dishonesty detectors to evaluate the *reliability of the most recent rating on a seller*. Provided the error bound of the trust model and environment vulnerability satisfy required conditions by Theorem 2, the computational trust model them would also be combined with the naive algorithm $\mathcal{N}$ to save cost.

The first tested computational trust model is the PeerTrust PSM/DTC algorithm proposed by Xiong and Liu (Xiong & Liu, 2004), which is presented earlier in Example 1 and henceforth mentioned as $\mathcal{L}$. With this algorithm, a peer i estimates the trustworthiness of another rater j based on the similarity between i's and j's ratings on some other sellers which both i and j have contacted with. The second model uses the maximum likelihood estimation-based learning algorithm as in Example 2 with a probabilistic decision rule. Given this computational model, abbreviated as $\mathcal{X}$, a peer i estimates the probability that a rater j is trustworthy so as to maximize the likelihood of getting the current set of ratings from i and j on those sellers with whom both i, j have experience. I also implemented a dishonesty detector $\mathcal{A}$ with very good misclassification errors: both α, β are less than $\varepsilon = 0.01$, and with a high cost of each time being used. In practice, a peer implements such an accurate detector by asking for information from the third party monitoring or consulting agents before buying things. The dishonesty detector $\mathcal{A}$ simulates a global computational trust model and is used to verify the relation between the learning accuracy and the cooperation level in the system with peers having the same input when learning the reliability of raters. The two algorithms $\mathcal{L}$ and $\mathcal{X}$ were used to test the efficiency of the peer-selection protocol in a more relaxed environment where peers used different algorithms with personalized inputs and settings to estimate rating behaviors. For the sake of readability, in the following experiments only the results for model $\mathcal{L}$ are shown, unless specified otherwise. The results for the model $\mathcal{X}$ are similar to those of $\mathcal{L}$, and the results of $\mathcal{A}$ are even better. The use of other global trust learning models like complaint-based algorithm (Aberer & Despotovic, 2001) or EigenTrust (Kamvar *et al.*, 2003) instead of the algorithm $\mathcal{A}$ is subject to future work.

As a base for other experiments, first the overall misclassification errors α, β of the computational trust models $\mathcal{L}$ and $\mathcal{X}$ under a variety of scenarios are estimated depending on the fraction of honestly reporting users h in the system. The most representative scenarios C_1, C_2, and C_3 for this experimentation are given in Table 4.2. These scenarios are designed based on the observation that it is not necessary to measure precisely the accuracy of a learning algorithm but only the overall trend and worst case misclassification errors α, β depending on the percentage of honest reporting peers h in the system. Furthermore, strategic selling behaviors could be excluded when estimating these misclassification errors for the reason that learning uses only historical data, and outcome of strategic selling behaviors can be assumed to follow one of the overall trends of these three extreme cases C_1, C_2, and C_3. These statistics are measured under three cases with different fractions of good and bad sellers in the systems (scenarios C_1, C_2, and C_3 in Table 4.2). Note that the accuracy of these algorithms have already been extensively studied in related work (Despotovic & Aberer, 2004b; Xiong & Liu, 2004), most of which having been shown to have low α, β under various attack scenarios. In this paper, the only difference is that such computational trust models are used to evaluate the reliability of the last rating on a seller, instead of estimating the trustworthiness of the seller. This reliability was determined based on reputation information of the user giving that very rating. Hence, the goal of the simulation here was to verify that misclassification errors of these trust models, when used in this new context, are still sufficiently low, so that the theoretical analysis on the relation between error bound and cooperation is applicable. That is, I need to verify that an approximate bound ε of the misclassification error rates α, β of the computational trust model being used satisfy the requirements to use Theorem 1, i.e., $\varepsilon < \varepsilon_{max}$ for a chosen threshold k. The obtained error bound ε would then be used as common knowledge in later experiments with various combination of strategic, bad, and good sellers.

Fig. 4.7 shows estimated maximal values of α, β of the learning algorithm $\mathcal{L}$ based on

PeerTrust PSM/DTC approach (Xiong & Liu, 2004), where $\mu_{trans} = 50$, and in three extreme cases C_1, C_2, C_3 of Table 4.2. Generally, the false positive errors α of $\mathfrak{L}$ are much lower where most peers staying in the system longer ($\mu_{trans} = 50$) and in less malicious environments (higher levels of honest reporters and good sellers), as we expected. With higher number of honest raters in the system, false negatives however can not be improved as much significantly as false positives. This may be due to my non-optimized implementation of the PeerTrust approach to estimating rating reliability, which could not learn from previous false negative errors in a noisy environment, e.g., good sellers may yield bad transaction outcomes from time to time. It is expected that in practice optimized implementations of such a model would be much more resilient and yield significantly lower false positives and false negative errors.

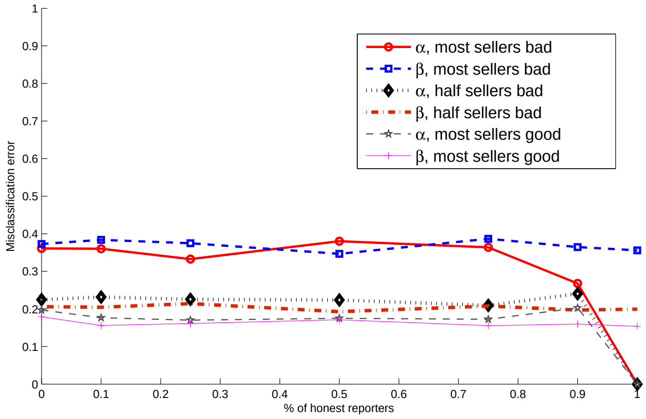
Nevertheless, even in the worst case of the simulation, $\max\{\alpha, \beta\}$ was well bellow 0.4, and thus there existed a threshold k for which Theorem 1 was applicable (Fig. 4.3 requires $k \geq 2$ in this case). However, it is more interesting to consider more representative cases with all different types of sellers, thus I mainly focused on the average case C_2 and C_3, where it may be approximated that $\max\{\alpha, \beta\} < \varepsilon = 0.25$. This error bound ε well satisfies the maximal error bound ε_{max} required by Theorem 1, with any threshold $k \geq 1$ (c.f. Fig. 4.3). These $max\{\alpha, \beta\}$ statistics were then used in our later experiments as the global knowledge ε of all peers. I also observed the same trend of these accuracy statistics in the presence of strategic peers in later experiments.

Different values of μ_{trans} are tested, for which I observed that when most sellers only participate in very few transactions ($\mu_{trans} < 10$), the trust learning algorithm could not collect enough historical data for accurate learning, resulting in high errors $\alpha, \beta > 0.5$ and thus using a trust model for learning the last rating was ineffective in enforcing cooperation.

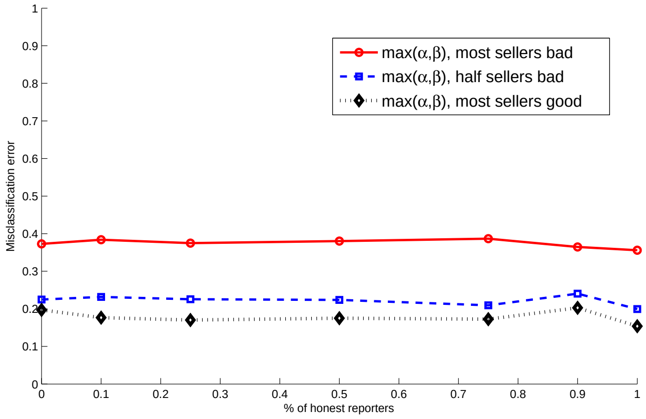
4.6.4 Cooperation in various environments

The levels of cooperation in the system with different types of sellers and raters are given in Fig. 4.8 for $\mu_{trans} = 50$ (cases C_4, C_5, C_6 of Table 4.2). The case of all strategic sellers shows even a better trend. Experiments are also performed in other extreme cases, e.g., all sellers, buyers, and raters strategic, yet the result was similar and thus are not given here. For small $\mu_{trans} < 10$ the cooperation dropped significantly, as the learning of rating reliability has no values at all.

When there are many types of rating and service behaviors (cases C_5, C_6 of Table 4.2), accumulated utilities of strategic peers, who follow serving strategies designed by the theoretical analysis, are also measured. Utilities of strategic sellers are close to those of good ones and significant higher than bad sellers. The reason is that strategic peers are enforced to cooperate most of their life. In fact, fully cooperative sellers have slightly better utilities. This was due to my pessimistic implementation of strategic peers, conjecturing that they needed not cooperate during their last Δ transaction. What happened during the simulation was that during these last Δ transaction, several strategic peers who cheated were also blacklisted. Therefore, under the proposed peer-selection mechanism it would be better off for those strategic peers to cooperate



(a)



(b)

Figure 4.7: Detailed misclassification errors of the PeerTrust model $\mathcal{L}$ used to evaluate the reliability of the last rating (a) and approximate error bound ε(b) in three scenarios C_1, C_2, C_3 of Table 4.2 with different fraction of bad sellers and malicious raters, $\mu_{trans} = 50$.

more even in their last transactions. The proposed approach still works even with dynamic joins and leaves of peers in the system, given that most sellers stay in the system long enough ($\mu_{trans} > 10$ in our simulation).

The impact of selecting the threshold k is shown in Fig. 4.9 (for cases C_4^k, C_6^k), which gives us the following observations. First, for a given ε, lower k means higher fraction of cooperation in the system (see Fig. 4.9 a), yet results in lower accumulative number of good transactions (Fig. 4.9b). Thus, with lower k values and higher ε, total number of good transactions in the system is lower due to the selective of buyers and some good buyers wrongly blacklisted. However, as the fraction of good transaction is high, the majority of those transactions taken

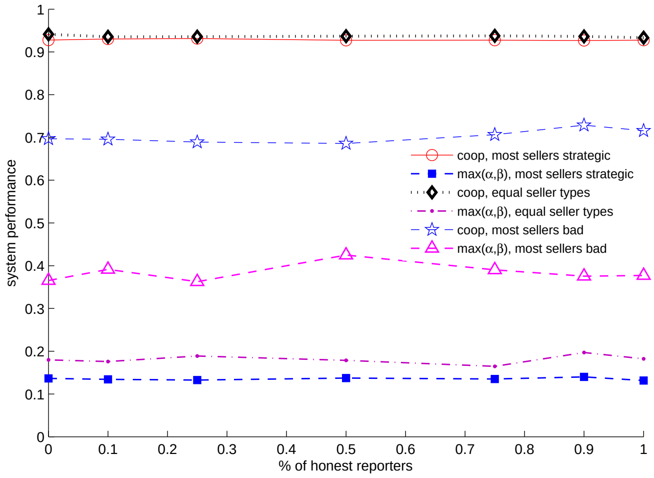
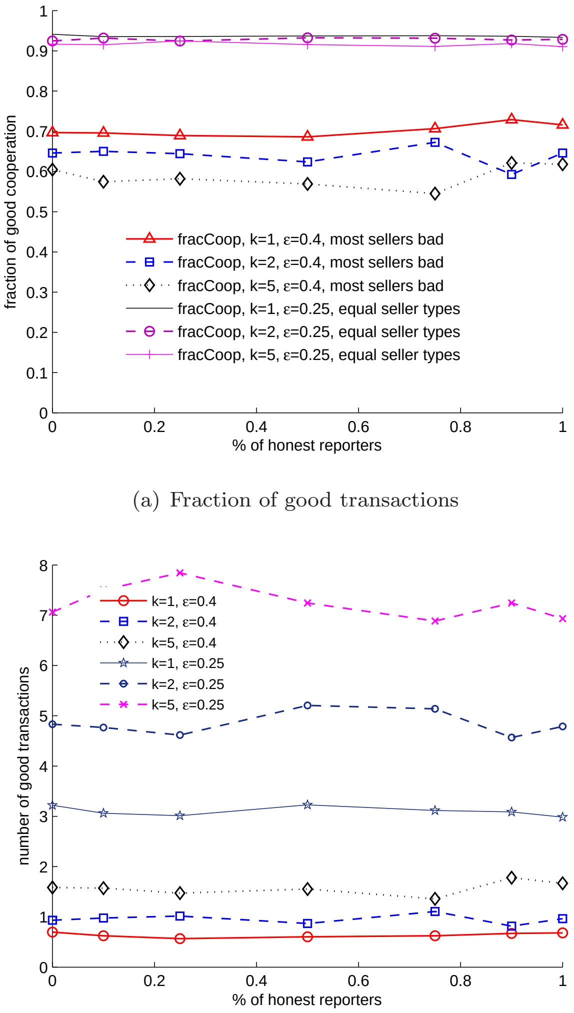


Figure 4.8: Relation between cooperation (fraction of good transactions) and accuracy of a trust learning model in environments with mixed behaviors: honest, malicious, and strategic. In two cases where most sellers strategic (C_5) and with approximately equal seller types (C_6) of and is very high since the learning accuracy is reasonably good. When most sellers are bad (C_4) fraction of good transaction is lower due to much less accurate evaluation of raters' behaviors.

place are successful. For the cases with small $\varepsilon = 0.25$, threshold k can be increased to accelerate the transaction rate in the system without worrying of selecting a bad provider and resulting in bad transactions, e.g., case $C_6^k, \epsilon = 0.25, k = 1, 2, 5$. Note that the measurement in Fig. 4.9(b) is an accumulated number of good transactions per peer. Such numbers would be increasing overtime, and hence their absolute values are less important. We sample it at the time where the simulation reached stationary regime only to compare the difference between the number of good transactions completed of different selection schemes with various k.

In case of all strategic peers C_7, we can also use a combination of two computational trust models to minimize the total learning cost (as in Theorem 2). Fig. 4.10 compares the cost of two approaches: the first one uses only the accurate yet expensive model (hypothetic) $\mathcal{A}$, the second uses $\mathcal{A}$ with a low probability. Therein, the cost of a computational trust model is measured as average number of sent and received messages for a good transaction. It is interesting to see that the learning cost can be reduced significantly by using the expensive model with a very low probability while still maintaining a high level of cooperation in the system. Furthermore, the approach combining two computational trust models learns faster and thus during the same simulation time, the total number of good transactions in the whole system (not shown in Fig. 4.10) is also much higher.



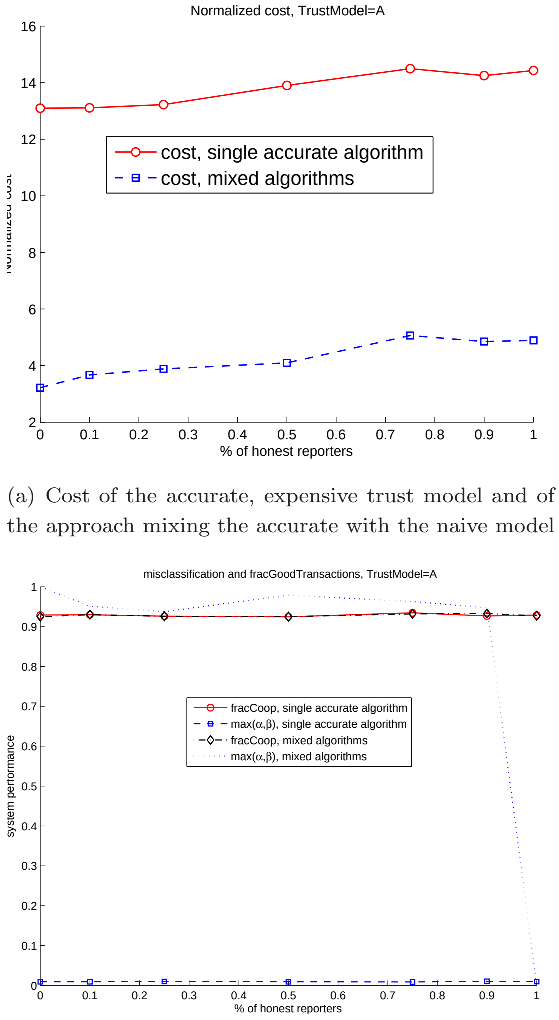
(a) Fraction of good transactions

(b) Number of good transactions per peer

Figure 4.9: Cooperation in the system with different ε and k. Results are given for scenarios C_4^k and C_6^k of Table 4.2.

4.7 The Issue of Cheap Pseudonyms and Possibilities of Deterring Whitewashing Behavior

So far, we have only considered the cases where it is difficult (or costly) for a participant to obtain a new identity and thus it is not optimal for a rational provider to simply cheat and rejoin under a new pseudonym without being punished. This section extends the work to the other cases where cheap identities are possible. We will study possible approaches of using computational trust models to enforce cooperation in scenarios where it is relatively easily for



(a) Cost of the accurate, expensive trust model and of the approach mixing the accurate with the naive model

(b) Misclassification errors and cooperation

Figure 4.10: Comparison the learning cost between two approaches: the use of a single accurate, expensive algorithm $\mathcal{A}$, and the mix of $\mathcal{A}$ and the naive algorithm $\mathcal{N}$ (a). Though both approaches have very high level cooperation (b), the second one that uses a mix of two different models has a significantly lower cost.

an existing provider to rejoin the system under a new identity, i.e., when an entrance cost is not easily imposed on participants. Hence, the sanctioning via global blacklisting as in Def. 6 has little impact on any provider with whitewashing behavior: such a provider may cheat and rejoin under a new identity, thus effectively avoiding the punishment.

I will prove that there exist dynamic pricing mechanisms to motivate cooperation of providers in all but some of their last transactions, even if cheap identities are possible. As soon as participants conform to these pricing mechanisms, cooperation is enforced even in the presence of strategic rating manipulation by malicious and rational players.

4.7.1 Identity premium-based pricing mechanisms

The key idea of the proposed approach is the use of an *identity premium function* (Def. 8). This premium allows a provider to sell its services, products, or resources at higher prices depending on the number of transactions it has completed in the system. Under this pricing scheme, an initially cheap identity (pseudonym) of a provider would have an increasingly significant value to the provider overtime. This implicit value of the current identity effectively eliminates the incentive of whitewashing for any rationally opportunistic providers.

Definition 8 *A provider who has finished $L \geq 0$ transactions in the system has a monotonically increasing* identity premium $f(L)$. *That is, every participant agrees that:*

- *a newly joined provider must sell services at some (low) price u_0, i.e., $f(0) = 0$.*
- *a provider having completed $L > 0$ transactions may sell services at a higher price of $u_0 + f(L)$, where $f(L) > 0$.*

The identity premium is similar to the notion of a reputation premium (Bhattacharjee & Goel, 2005) yet simpler to verify, since the latter requires that the reliability of a reputation value is estimated and included in the analysis. An identity premium can also be readily implemented in different ways beside using a direct pricing mechanism. For example, a system may use a ranking function s.t. a long-staying provider is matched with more clients and thus gains more. This implementation specially applies to the case where providers offer non-depleting resources, i.e., the same resource can be sold to many clients at the same cost as to a single client.

We will study the properties of the identity premium function $f(L)$ in relation to the initial price u_0 to ensure that rational providers are motivated to cooperate in most of their transactions, *irrespective of the cost of cheap pseudonyms*, and in presence of *strategic or malicious manipulation of ratings* by participants.

To this end, a first step is to use the identity premium function $f(.)$ to specify a pricing mechanism as in Def. 9.

Definition 9 *Consider a service with an original price $u_* \leq u \leq u^*$, which is publicly known. A provider who has completed $L \geq 0$ transactions in the system sells this service at the price determined by the following pricing scheme $\underline{P}(\phi, f)$:*

$$\underline{P}(\phi, f) = u(1 - \phi) + f(L) \tag{4.5}$$

where $f(L)$ is the identity premium of the provider. At $L = 0$, or at zero identity premium, the provider has to sell a service at an initial price $u(1 - \phi) + f(0) = u(1 - \phi) \ll u$. This price is determined by a parameter $0 < \phi < 1$, which may depend on the original price u.

Apparently, under the pricing scheme $\underline{P}(\phi, f)$, a new provider must sell a service at a much lower price $u(1 - \phi)$ than the actual value u of the service. A provider staying in the system for many transactions may sell services and resources at higher prices. Being able to stay in the system and sell services at higher prices in later transactions compensates the loss of the providers during earlier transactions where they have zero or small identity premiums. Therefore, even if establishing new identities are relatively cheap, it is expected that every rational provider finds it optimal to keep the same identity for the whole life-time rather than cheating, leaving, and joining with new identity. In other words, whitewashing is not optimal for any rational provider.

Given a bound ε for α, β of the dishonesty detector being used by peers in the system, Theorem 3 shows the relation between the error bound ε and its effectiveness in enforcing cooperation of a provider during its life-time in the case with possible cheap pseudonyms.

Theorem 3 *Given the peer-selection protocol $\mathcal{S}_k = \langle \mathcal{R}, k \rangle$ where the dishonesty detector $\mathcal{R}$ has misclassification errors α, β upper-bounded by $\varepsilon < 0.5$.*

Consider any rational provider with N services to sell. Let $u_* \leq u_i \leq u^*, i = 1, ..., N$ be the original prices of services sold by the provider in the i-th transaction. Suppose that the pricing scheme $\underline{P}(\phi, f)$ is used. The cheating gain when selling a service of price u_i is assumed to be γu_i, where $0 < \gamma \leq 1$ does not depend on u_i. Define $\lambda = \frac{\gamma}{(1-\varepsilon)^k - \varepsilon^k}$.

(i) If u_is are different and $\lambda > 1$, the following identity premium ensures that cooperation is always the best response strategy of the provider in any *transaction $i = 1, ..., N-1$, for any $0 < \phi_i < 1$:*

$$f(L) \quad = \quad \sum_{i=1}^{L} \lambda^{L-i+1} u_i (1 - \phi_i) \tag{4.6}$$

(ii) If $u_i = 1, i = 1, ..., N$, or providers sell services of similar original values, with any $\lambda > 0$ the following identity premium function is sufficient to enforce cooperation for a provider in selling every but the last service:

$$f(L) \quad = \quad \sum_{i=1}^{L} \lambda^{L-i+1}(1 - \phi) = (1 - \phi)\lambda \frac{1 - \lambda^L}{1 - \lambda} \tag{4.7}$$

The above results hold even in presence of strategic manipulation of ratings by peers and even if cheap pseudonyms are possible.

Proof 7 *We first prove (i). A rational provider apparently does not have incentives to cooperate when selling the last service. Consider a rational provider who still has at least $\Delta > 0$ services to sell after it finishes selling the current one.*

Suppose that the provider has finished $L - 1 \geq 0$ transactions with the current identity and gained an utility of $U > 0$. In the current L-th transaction, the provider has an identity premium of $f(L-1)$. Let $u_* \leq u_L \leq u^*$ be the actual price of the current service, the provider may sell the service at a price $u_L(1 - \phi) + f(L-1)$.

Denote as $U(L, \Delta)$ the best (maximized) expected utilities a provider with an identity premium $f(L)$ may get for the remaining Δ transactions. $U(0, \Delta)$ thus corresponds to the case *when the provider has no identity premium: e.g., a newly joined provider or a provider who just left and then re-joined under a new identity.*

By similar reasoning as that of Theorem 1, the probability that the provider is globally blacklisted after the current transaction is some $x_b^k \leq \varepsilon^k$. *This inequality holds even in presence of malicious or strategic manipulation of ratings by providers, provided that misclassification errors* α, β *of* $\mathcal{R}$ *are less than* ε. *A globally blacklisted provider with more services to sell has to join the system under a new identity with a zero identity premium. On the other hand, the probability the provider is globally blacklisted is some* $y_b^k \geq (1-\varepsilon)^k$.

Let U_{honest} *[resp.* U_{cheat}*] as the best expected life-time utilities of the provider if she is honest [resp. cheating] in the current transaction, it follows that:*

$$\begin{aligned}
U_{honest} &= U + u_L(1-\phi) + f(L-1) + (1-x_b^k)U(L,\Delta-1) + x_b^k U(0,\Delta-1) \\
U_{cheat} &= U + [u_L(1-\phi) + f(L-1)](1+\gamma) + (1-y_b^k)U(L,\Delta-1) + y_b^k U(0,\Delta-1) \\
\Rightarrow \delta_{hc} &= U_{honest} - U_{cheat} = -[u_L(1-\phi)+f(L-1)]\gamma + (y_b^k - x_b^k)(U(L,\Delta-1) - U(0,\Delta-1)) \\
&\geq -[u_L(1-\phi)+f(L-1)]\gamma + ((1-\varepsilon)^k - \varepsilon^k)(U(L,\Delta-1) - U(0,\Delta-1)) \qquad (4.8)
\end{aligned}$$

Suppose that in the next transaction the provider sells another service whose original price is u'. *If the provider is honest in the next transaction, we have:*

$$\begin{aligned}
U(L,\Delta-1) &= u'(1-\phi') + f(L) + (1-x_b^k)U(L+1,\Delta-2) + x_b^k U(0,\Delta-2) \\
U(0,\Delta-1) &= u'(1-\phi') + (1-x_b^k)U(1,\Delta-2) + x_b^k U(0,\Delta-2) \\
U(L,\Delta-1) - U(0,\Delta-1) &= f(L) + (1-x_b^k)(U(L+1,\Delta-2) - U(1,\Delta-2)) \qquad (4.9)
\end{aligned}$$

On the other hand, if the provider is cheating in the next transaction:

$$\begin{aligned}
U(L,\Delta-1) &= [u'(1-\phi')](1+\gamma) + f(L) + (1-y_b^k)U(L+1,\Delta-2) + y_b^k U(0,\Delta-2) \\
U(0,\Delta-1) &= [u'(1-\phi')](1+\gamma) + f(0)(1-y_b^k)U(1,\Delta-2) + y_b^k U(0,\Delta-2) \\
U(L,\Delta-1) - U(0,\Delta-1) &= f(L) + (1-y_b^k)(U(L+1,\Delta-2) - U(1,\Delta-2)) \qquad (4.10)
\end{aligned}$$

From Equations (4.9,4.10), and note that $x_b^k \leq \varepsilon^k \leq (1-\varepsilon)^k \leq y_b^k \leq 1$, *we have:*

$$\begin{aligned}
U(L,\Delta-1) - U(0,\Delta-1) &\geq f(L) - f(0) + \min(1-y_b^k, 1-x_b^k)(U(L+1,\Delta-2) - U(1,\Delta-2)) \\
\Rightarrow U(L,\Delta-1) - U(0,\Delta-1) &\geq f(L) + (1-y_b^k)(U(L+1,\Delta-2) - U(1,\Delta-2)) \qquad (4.11)
\end{aligned}$$

By similar reasoning the following recurrence relation can be found for $1 \leq i \leq \Delta - 2$:

$$U(L+i,\Delta-i-1) - U(i,\Delta-i-1) \geq f(L+i) - f(i) + (1-y_b^k)(U(L+i+1,\Delta-i-2) - U(i+1,\Delta-i-2)) \qquad (4.12)$$

$$\text{and } U(L+\Delta,0) - U(\Delta-1,0) = f(L+\Delta) - f(\Delta-1) \qquad (4.13)$$

From the recurrence relations of Equations (4.11,4.12,4.13) we find[1]*:*

$$U(L,\Delta-1) - U(0,\Delta-1) \geq f(L) + \sum_{i=1}^{\Delta-1}(1-y_b^k)^i(f(L+i) - f(i)) \qquad (4.14)$$

[1] Rigorously, the probabilities x_b^k, y_b^k may be different in each equation (4.12), and thus in (4.14), y_b^k shall be the largest one among these y_b^k. However, to simplify the notation we will ignore such differences.

Let $f(L)$ be a monotonically increasing function of L, then $f(L+i) - f(i) \geq 0$. From Equations (4.8,4.14), it follows that:

$$\delta_{hc} \quad = \quad U_{honest} - U_{cheat} \geq -[u_L(1-\phi) + f(L-1)]\gamma + ((1-\varepsilon)^k - \varepsilon^k)f(L)$$

Cooperation in this (L-th) transaction is a dominant strategy for the provider iff $\delta_{hc} \geq 0$, or:

$$f(L) - \frac{\gamma}{(1-\varepsilon)^k - \varepsilon^k} f(L-1) \geq \frac{u_L(1-\phi)\gamma}{(1-\varepsilon)^k - \varepsilon^k} \tag{4.15}$$

Denote $\lambda = \frac{\gamma}{(1-\varepsilon)^k - \varepsilon^k} > 0$. By similar reasoning, cooperation is the dominant strategy in all transactions $1, ..., L$ iff:

$$\begin{aligned}
f(L) - \lambda f(L-1) &\geq u_L(1-\phi_L)\lambda && (4.16)\\
f(L-1) - \lambda f(L-2) &\geq u_{L-1}(1-\phi_{L-1})\lambda && (4.17)\\
&\dots\\
f(2) - \lambda f(1) &\geq u_2(1-\phi_2)\lambda && (4.18)\\
f(1) &\geq u_1(1-\phi_1)\lambda, \text{ where } f(0) = 0 && (4.19)\\
\Rightarrow f(L) &\geq \sum_{i=1}^{L} \lambda^{L-i+1} u_i(1-\phi_i) && (4.20)
\end{aligned}$$

The following identity premium function satisfies every above constraint:

$$f(L) \quad = \quad \sum_{i=1}^{L} \lambda^{L-i+1} u_i(1-\phi_i) \tag{4.21}$$

Furthermore, this identity premium is an increasing function of L with any $\lambda > 1$, hence the requirement $f(L+i) > f(i)$ in (4.14) is met.

Therefore, provided that the client chooses to buy the service, cooperation is always the best response strategy of the provider in any transaction after which the provider still has $\Delta = N - L > 0$ services to sell. In other words the provider is motivated to cooperate at every transaction $L = 1, ..., N-1$ and thus (i) is proved. For the simple case where all $u_i = 1, i = 1, ..., N$, the requirement of $\lambda > 1$ is unnecessary, which means the proof of (ii) follows immediately.

The identity premium $f(L)$ determines the additional amount the client must pay to motivate the provider. This cost of by the client to motivate cooperation of rational providers in the system is an inevitable cost in any open system where cheap identities are possible. Apparently, a smaller identity premium $f(L)$ results in lower prices and thus is more encouraging participation of clients. Regarding the constraint of (4.20) in the analysis, the $f(L)$ defined in Theorem 3 is the smallest possible identity premium.

Providers, on the other hand, need to sell their first services at very low prices to gain identity premium before selling other services at higher prices. If providers have many services to sell, eventually they would be able to sell at very high price thanks to their gained identity premiums.

In general small values of $\lambda = \frac{\gamma}{(1-\varepsilon)^k - \varepsilon^k} < 1$ are preferable, such that eventually providers do not get a so much high price, as from (4.21). The parameter λ depends on the nature of the problem domain $0 < \gamma = \frac{gain}{price} \leq 1$ and the system design parameters ε, k.

For example, in eBay-like systems, cheating means that the provider (seller) may gain the whole price paid to the article, $\gamma = 1$ and $\lambda = 1/((1-\varepsilon)^k - \varepsilon^k) > 1$ for any $k > 0, \varepsilon < 0.5$. This means that a completely open (i.e., with cheap identities) and decentralized version of an eBay-like system is likely impractical if everyone is rational: buyers must have to pay very high prices to ensure the cooperation of the sellers, otherwise sellers simply cheat to gain more and then whitewash the bad behavior. In another example of service provisioning, providing bad services has a lower but non-zero value to the client, it is acceptable to assume that $\gamma < 1$. Given the availability of a sufficiently accurate trust learning mechanism with small ε and with appropriate k, it is still possible that $\lambda < 1$.

With $\lambda > 1$, the price of services ad infinitum can not be bounded and depends on the number of services the provider wants to sell during its whole life-time. With $\lambda < 1$, it is possible to design a pricing mechanism based on the identity premium concept so that the price ad infinitum $L \to \infty$ is bounded as follows.

If $u_i = 1, i = 1, ..., N$, as in item (ii) of Theorem 3, it is clear that $\lim_{L\to\infty} f(L) = \frac{\lambda(1-\phi)}{1-\lambda}$, thus the price ad infinitum is naturally bounded. A similar result can be obtained even if u_is are different. Noting that $u_i \leq u^*, i = 1, ..., N$. Use the same $\phi = \phi_i$ for $i = 1, ..., N$, then an alternative for the identity premium in (4.21) is:

$$f(L) = u^*(1-\phi)\sum_{i=1}^{L}\lambda^{L-i+1} = u^*(1-\phi)\lambda\frac{1-\lambda^L}{1-\lambda} \tag{4.22}$$

$$\Rightarrow \lim_{L\to\infty} f(L) = \frac{u^*\lambda(1-\phi)}{1-\lambda} \tag{4.23}$$

With this identity premium the price ad infinitum is also bounded by $1-\phi+\frac{u^*\lambda(1-\phi)}{1-\lambda}$, for $\lambda < 1$.

Understanding identity-premium as probability of defection: with $\lambda < 1$, let us rewrite the price $\underline{P}(\phi, f)$ as:

$$\begin{aligned}
\underline{P}(\phi, f) &= u(1-\phi) + f(L) = u[1 - (\phi - \frac{f(L)}{u})] \\
\lim_{L\to\infty}\frac{f(L)}{u} &= \lim_{L\to\infty}\frac{u^*\lambda(1-\phi)(1-\lambda^L)}{u(1-\lambda)} = \frac{u^*\lambda(1-\phi)}{u(1-\lambda)} \\
0 < \phi - \frac{f(L)}{u} \leq 1 &\Leftrightarrow \lim_{L\to\infty}\frac{f(L)}{u} \leq \phi \\
\Leftrightarrow \phi &\geq \phi_{min} = \frac{\frac{u^*\lambda}{u(1-\lambda)}}{1+\frac{u^*\lambda}{u(1-\lambda)}} = \frac{1}{1+\frac{u(1-\lambda)}{u^*\lambda}} \\
0 < \phi_{min} < 1 &\Leftrightarrow \lambda < 1
\end{aligned}$$

Therefore, with $\lambda < 1$ and $1 > \phi \geq \phi_{min} > 0$, $\phi - f(L)$ can be interpreted as the probability the client believes the provider would defect in the current transaction, and the price of the service being transacted is determined based this defection probability. Under the proposed identity premium-based pricing framework, the probability of defection is considered as smaller for those providers having stayed longer in the system.

In the next section, we will study whether such the use of an identity premium concept brings significant advantages or disadvantages to the providers and based on that to find an optimal setting for the pricing mechanism $\underline{P}(\phi, f)$ to minimize these (dis) advantages.

4.7.2 Inefficiency of an identity premium-based pricing mechanism

Theorem 3 puts a requirement on the shape of the identity premium function $f(.)$, yet it places no special requirements on the parameter ϕ that decides the initial price of a service.

Under the pricing scheme $\underline{P}(\phi, f)$, a provider initially must sell a service at some price lower than its actual value, and increase the price over time to compensate for previous losses. It is important to understand whether a provider may gains or loses a significant amount from such a pricing scheme. Compared to a usual system not employing the identity-premium concept, the additional benefit $g(N)$ that the pricing scheme $\underline{P}(\phi, f)$ gives a provider with N services to sell is defined as:

$$g(N) = \sum_{i=1}^{N}[u_i(1-\phi_i) + f(i-1)] - \sum_{i=1}^{N} u_i = \sum_{i=1}^{N}(f(i-1) - u_i\phi_i) \tag{4.24}$$

where $u_* \le u_i \le u^*$ is the actual price of the service sold in i-th transaction, ϕ_i is the parameter deciding the initial price at zero identity premium, and $f(i)$ is the identity premium of the i-th transaction.

We formally define such additional (dis)advantages, or sum of gains or losses of all providers as the efficiency of the pricing scheme in a system (Def. 10).

Definition 10 *The inefficiency of a system with an identity premium-based pricing mechanism is define as $\sum_s g_s(N)$, where $g_s(N) = \sum_{i=1}^{N}(f(i-1) - u_i\phi_i)$ is the individual (dis)advantage of a provider s.*

Depending on the nature of the problem and the sequence of services sold by a provider, the total gain $g(N)$ of the provider from his identity premium can be positive or negative. If $g(N) > 0$, in the current system clients pay higher prices for services compared to a system without an identity premium-based pricing mechanism. As a result, higher values $g(N)$ may deter clients from participating in the system, as services are generally more expensive. On the other hand if $g(N) < 0$, providers must accept selling services at lower prices compared to normal systems where no identity premiums are used. Negative values of $g(N)$ may deter participation of providers since they generally get less revenue.

Thus collectively, the average gain $g(N)$ reflects the inefficiency of the system using an identity premium-based pricing scheme $\underline{P}(\phi, f)$ in encouraging participation of clients and providers. $g(N)$ is the cost inherent to the problem of cheap identities and unfortunately can not be further reduced as it does not rely much on system wide setting, e.g., ϕ_i, but on other domain specific variables, such as the values u_i of the services and the corresponding gains v_i by cheating.

Theorem 4 *Consider a simple case where providers sell services of the same original price of 1. We have:*

- *For any provider with N services to sell and N is known, the identity premium-based pricing mechanism has no inefficiency to the system w.r.t this provider, i.e., $g(N) = 0$, if the initial price is set at $1 - \phi$ where:*

$$\phi = \frac{\lambda^{N+1} - N\lambda^2 + (N-1)\lambda}{\lambda^{N+1} - (N+1)\lambda + N} \tag{4.25}$$

- *If all providers have infinitely many services to sell and $\lambda < 1$, the identity premium-based pricing mechanism has a small bounded inefficiency of $-\frac{\lambda}{1-\lambda}$ to each provider if the initial price for a service is $1 - \lambda$.*

Proof 8 *Since $u_i = 1, i = 1, ..., N$, we have a simplified identity premium function:*

$$f(L) = \sum_{i=1}^{L} \lambda^{L-i+1}(1-\phi) = (1-\phi)\lambda\frac{1-\lambda^L}{1-\lambda} \tag{4.26}$$

The inefficiency of the pricing mechanism to the provider is:

$$g(N) = \sum_{i=1}^{N}(f(i-1) - \phi) = \sum_{i=1}^{N}((1-\phi)\lambda\frac{1-\lambda^{i-1}}{1-\lambda} - \phi) \tag{4.27}$$

$$= N\frac{\lambda(1-\phi) - \phi(1-\lambda)}{1-\lambda} - \frac{\lambda(1-\phi)(1-\lambda^N)}{(1-\lambda)^2} \tag{4.28}$$

$$= \frac{\lambda^{N+1} - N\lambda^2 + (N-1)\lambda - \phi(\lambda^{N+1} - (N+1)\lambda + N)}{(1-\lambda)^2} \tag{4.29}$$

$$g(N) = 0 \quad \Leftrightarrow \quad \phi = \frac{\lambda^{N+1} - N\lambda^2 + (N-1)\lambda}{\lambda^{N+1} - (N+1)\lambda + N} = \frac{A}{B} \tag{4.30}$$

It is observed that for any $\lambda > 0$:

$$A = \lambda^{N+1} - N\lambda^2 + (N-1)\lambda = N\lambda(1-\lambda) - \lambda(1-\lambda^N) \tag{4.31}$$

$$= \lambda(1-\lambda)[N - (1 + \lambda + ... + \lambda^{N-1})] > 0 \tag{4.32}$$

$$B = \lambda^{N+1} - (N+1)\lambda + N = N(1-\lambda) - \lambda(1-\lambda^N) \tag{4.33}$$

$$= (1-\lambda)[N - \lambda(1 + \lambda + ... + \lambda^{N-1})] > 0 \tag{4.34}$$

$$\text{and } A - B = -N(1-\lambda)^2 < 0 \tag{4.35}$$

Therefore, with any $\lambda > 0$, in (4.30) we always have $0 < \phi < 1$.

For $N \to \infty$ and $\lambda < 1$, it follows that $\phi \to \frac{\lambda - \lambda^2}{1-\lambda} = \lambda$, and the initial service price is $1 - \lambda$. The price ad infinitum in this case is $1 - \lambda + \lambda(1 - \lambda^L) \to 1$. We can verify that for this specific setting, the inefficiency of each provider is $\lim_{N\to\infty} g(N) = -\lim_{N\to\infty} \frac{\lambda(1-\lambda^N)}{1-\lambda} = -\frac{\lambda}{1-\lambda}$, thus the theorem holds naturally.

4.7.2.1 Considering the risks of clients

It is also important to estimate and compare the risk of a client when participating in two systems: one using an identity premium-based pricing mechanism, and one without such pricing scheme. By convention, the risk of a client in buying a service with a price p and with a probability c that the provider cheats in the transaction is defined as pc.

We consider the following two cases. First, if providers are rational and sell infinitely many services, according to Theorem 3, the probability that a provider (in a system using an identity-premium) cheats in any transaction is zero. Thus the risk of a client when participating in a system with identity premium-based pricing is also zero, irrespectively of the price of the service. This is true for every client in any transaction and for any $\lambda > 0$. The price may be very large for $\lambda > 1$ or may be bounded if $\lambda < 1$. In the former case, the application of an identity premium-based concept may be not preferable as high prices would naturally deter the participation of potential clients.

In the second case, suppose that each provider sells a limited number of services, and the distribution of number of services N of providers is known, with a cumulative distribution function $F(N)$. From Theorem 3, in a system with price determined based on identity premium, a rational provider would only defect in selling the very last service. Consider the transaction on a service with original price u offered by a provider who has finished L transactions. The probability this provider cheats would be the probability that this provider has no more services to offer after the current transaction, which is:

$$c \quad = \quad 1 - F(L+1)$$

The risk of the client in transacting with this provider is thus:

$$\begin{aligned} risk_1(u,L) &= u(1-\phi+\frac{f(L)}{u})(1-F(L+1)) \\ &= u(1-F(L+1))[1-\phi+\frac{u^*\lambda(1-\phi)(1-\lambda^L)}{u(1-\lambda)}] \\ &= u(1-F(L+1))(1-\phi)\frac{\lambda^{L+1}\frac{u^*}{u}-\lambda(\frac{u^*}{u}-1)-1}{\lambda-1} \end{aligned}$$

On the other hand, in a system without identity premiums and with possible cheap identities, anytime the provider may defect and rejoin with new identities without being punished. All other factors[1] being equal, the risk of the client would be:

$$risk_2(u,L) = u.1 = u$$

We compare the two risks $risk_1(u,L)$ and $risk_2(u,L)$ in two following cases. First, as proven in Section 4.7.1, for $\lambda < 1$ and $\phi \geq \phi_{min} = \frac{1}{1+\frac{u(1-\lambda)}{u^*\lambda}}$, it is apparent that $1-\phi+\frac{f(L)}{u} < 1$. It follows that with any $L > 0, risk_1(u,L) < u = risk_2(u,L)$. In other words, the system with identity premium is always better for the clients in terms of minimizing the risk of being cheated when $\lambda < 1$.

[1] such as the rationality of the provider and his malicious intention

Second, for $\lambda > 1$, one may verify that there always exists some $L_{min} > 0$ such that for $L > L_{min}$, $risk_1(u, L) > risk_2(u, L)$. In this case, it is more risky for client to buy services from providers with larger L compared to a system not using an identity premium concept. This is intuitive since while their services are priced more highly, it is still more likely that these providers may finish trading very soon.

In summary, a system with identity premium may be much better for clients in terms of minimizing their risks under certain conditions, namely in those application scenarios with $\lambda < 1$, or where providers have infinitely many services to sell to their potential clients.

4.8 Related Work

A large number of works in the trust research literature focus on developing appropriate computational models for learning peer behaviors based on their historical performance information, for example (Kamvar *et al.*, 2003; Levien, 2002; Srivatsa *et al.*, 2005; Xiong & Liu, 2004). Comparative surveys of research literature on this subject can be found elsewhere (Dellarocas, 2005b; Despotovic & Aberer, 2006; Golbeck, 2006; Jøsang *et al.*, 2007). These works are complementary to the work in this chapter, since any robust and accurate computational learning models can be used in the reputation-based peer selection protocol.

This work is inspired by the analysis of Dellarocas (Dellarocas, 2005a), which studies various design parameters of reputation mechanisms. Some conclusions in the work of Dellarocas (Dellarocas, 2005a) coincide with the analysis in this chapter, e.g., the robustness of the naive trust model algorithm (Section 4.3.3). However, this work has considered many more aspects of a reputation-based computational trust model, namely its accuracy and cost, from which I propose an approach to use such model in an incentive-compatible and cost-efficient way, as well as performed empirical experimental studies of reputation effects on strategic agents in decentralized and dynamic environments.

The cost-efficient reputation management solution in Section 4.4 is an application of inspection game theory (Avenhaus *et al.*, 2002), wherein an accurate yet expensive trust learning algorithm plays the role of an inspector detecting dishonest behaviors of sellers who take the role of an inspectee of the game. The provided results confirm the important role of an effective identity management scheme, as earlier identified by Friedman and Resnick (Friedman & Resnick, 2001). To the best of my knowledge, this work is the first work to apply inspection game theory in reputation systems. The most related work to the proposed cost-effective reputation management approach is (Agrawal & Terzi, 2006), yet it addresses another problem of how to control the behaviors of agents in a centralized sovereign information sharing scenario. They propose to use an auditing device as a trusted centralized agent and decide the appropriate frequency of auditing. The notion of punishment amount and frequency are similar to the notion of probability to use the accurate and expensive computational trust models in the peer-selection protocol of this chapter. My work, on the contrary, studies the possibility of using computational trust mechanisms in a cost-effective way and provides general results applicable to a wider range of applications with different degrees of centralization.

4.9 Discussion and Conclusion

Many interesting observations can be derived from this analysis. First of all, by showing that there exist certain sufficient conditions for a computational trust model to be effective in enforcing cooperation, this work provides an initial positive answer to the question whether existing computational trust models in the literature can be used effectively to enforce cooperation in environments where both malicious and rational behaviors are present. It also implies that in a heterogeneous environment where peers use different learning algorithms with certain accuracy to learn trustworthiness of their potential partners, cooperation also emerges.

Secondly, depending on the presence of rationality from participants of the scenario being studied, either simple or sophisticated trust learning algorithms are appropriate: in environments with fully rational peers caring about their utilities, simple trust models are sufficient to stimulate cooperation in the system. In other scenarios where bad peers have the only goal of bringing the system down at any cost, sophisticated learning algorithms are necessary to filter out malicious agents.

The presented peer-selection protocol is based on the idea to use a computational trust model to evaluate the reliability of a most recent report on a seller. Such a model plays the role of an inspector to detect only *past* malicious behaviors of raters and sellers. The introduction of such a simple selection protocol that punishes bad peers creates a social control mechanism to stimulate cooperation of rational peers in the next transactions. This is different from other systems that mainly use sophisticated learning algorithms to predict future peer performance from the past. The inclusion of well-tested learning algorithms with high accuracy to filter out malicious partners also helps the system to perform well even in mixed environments with various malicious and rational behaviors. To a larger extent, the presented protocol establishes an umbrella framework to use reputation information effectively by exploiting both its sanctioning and signaling roles (Dellarocas, 2005a) in decentralized and self-organized systems.

The theoretical result of this chapter proves that the key to ensure cooperation is not the accuracy/errors of the trust learning algorithm being used, but on the management of identities: establishing a new identity must be costly so that peers want to stay in the system for many transactions rather than changing their identities and start over. Such whitewashing behaviors can be prevented by using many existing techniques, for example, to impose an entrant fee on newly joined peers. For those applications where peers may get very large temporary gains for cheating, it is very hard to design any totally decentralized incentive-based mechanism to motivate short-term peers to be fully cooperative, and reputation information may not be an effective tool anyway.

I have also analyzed possibilities of enforcing cooperation in the case where cheap identities are possible. This analysis reveals that for certain applications where cheating gains are bounded, the problem of cheap identities can be solved effectively. Under certain conditions, a pricing mechanism based on the notion of an identity premium is beneficial to both providers and clients in the system. Such an identity premium-based pricing mechanism ensures that rational providers are better-off cooperative in all but their very last transaction, even in presence of possible strategic manipulation of ratings and the cheap cost of identities. This is the very first work on solving the problem of cheap pseudonyms by using decentralized trust management mechanisms in practical application scenarios.

As future work, it is my interest to derive some theoretical bounds on the cooperation level in the system using the proposed reputation management approach, given arrival and up-time distributions of peers. An implementation of various serving strategies of bounded rational peers in the simulation framework may be also necessary. Such empirical simulations may give us more insights to the effectiveness of different trust learning algorithms in enforcing cooperation and building trust in presence of bounded-rational peers with limited available information. At the end of the day, this analytical and simulation approach would help us to provide a "cookbook" for the application of different trust learning algorithms in open and decentralized systems.

Chapter 5

Applications of Computational Trust Models

This chapter presents two practical applications of computational trust models in two different areas. In the first application, I propose using a reputation-based trust management approach to enable better selection and ranking of (Web) services in a service-oriented architecture based on quality of the services (QoS). In this approach, ratings on different service quality attributes are collected from many users, while trust management techniques are applied to select and use only reports of reputable users in the quality evaluation.

The second application of computational trust models presented in this chapter is the use of an appropriate trust measure to evaluate and select most potentially reliable delegates in a key backup and recovery scenario in a distributed social network. Therein the service provided by each peer is the storing of a share of some secret keys, and the trustworthiness of a peer is well correlated to its reliability in keeping those shares secretly.

The work in this chapter is published in (Vu *et al.*, 2005a, 2009a). The QoS-based service selection and ranking algorithm described in this chapter is a part of my other work on Semantic Web service discovery in framework of the European research project DIP (dip, 2004), which is published elsewhere (Vu *et al.*, 2005b, 2006, 2007)

5.1 QoS-based Service Ranking and Selection

QoS-based service selection mechanisms will play an essential role in service-oriented architectures, as e-Business applications want to use services that most accurately meet their requirements. Standard approaches in this field typically are based on the prediction of services' performance from the quality advertised by providers as well as from feedback of users on the actual levels of QoS delivered to them. The key issue in this setting is to detect and deal with false ratings by dishonest providers and users, which has only received limited attention so far[1]. This work presents a new QoS-based selection and ranking for online services, using a novel technique of trust and reputation management to filter out biased reports. Specifically, the proposed approach relies on a number of reports from trusted agents as reference reports and additionally use clustering to identify group of related raters. Only quality ratings proven to be reliable are included in the quality evaluation of the services. I give a formal description of the approach and validate it with experiments which demonstrate that the solution yields high-quality results under various realistic cheating behaviors.

5.1.1 Introduction

Practical applications of service oriented architectures (SOA) are ubiquitous and have grown much beyond organizational and application domain boundaries. Among prominent examples are workflow management systems (wmf, 2008), e-Science applications on the Grid (Foster & Kesselman, 2003), and computing infrastructures for commercial sites on the Cloud computing platform (Varia, 2008).

While provisioning of services has to take place electronically or traditionally depending on the service application domain, eventually any service can be described, discovered and requested electronically via the Internet. Therefore, most of them can be effectively represented by associated Web services. Among them, services can be described, requested and executed over the Internet are becoming an essential part of business applications. For instance, several commercial systems have built their computing infrastructures using available data storage services (ama, 2008), software-on-demand (salesforce.com), and online computational services (Google App Engine, Amazon EC2). Examples of traditional services that can be published as Web services are even more numerous, from hotel booking, travel planning to house building or accommodation services (O'Sullivan *et al.*, 2005).

Quality of service (QoS) is of paramount importance in SOA environments. Thanks to its loosely coupled properties, composing services of an SOA-based system can be implemented by different (and possibly unknown) providers. Therefore, it is highly important to assure that quality of those services to be integrated in the system is satisfactory.

In business scenarios with competitive providers, QoS is amongst the most decisive criteria influencing a user in the selection of a certain service among several functionally equivalent ones. Thus it is the key to a provider's business success. Therein a key issue is to enable e-Business applications to discover services which best meet their requirements in terms of quality, i.e., performance, throughput, reliability, availability, trust, etc. Thus QoS-based Web service selection and ranking mechanisms will play an essential role in service-oriented architectures, especially when the semantic matchmaking process returns lots of services with comparable functionalities.

This work presents a QoS-based Web service selection and ranking approach which uses trust

[1] This work, published in (Vu *et al.*, 2005a), is among the earliest work using trust management in quality-based service discovery and has received numerous citations since then.

and reputation evaluation techniques to predict the future quality of a service. The work is based on requirements from a real-world case study on virtual Internet service providers (VISP) from an industrial partner in one of research projects in my lab (http://dip.semanticweb.org/). In a nutshell, the idea behind the VISP business model is that Internet Service Providers (ISPs) describe their services as Semantic Web services (Berners-Lee *et al.*, 2001), including QoS such as availability, acceptable response time, throughput, etc., and a company interested in providing Internet access, i.e., becoming a VISP, can look for its desired combination of services taking into account its QoS and budgeting requirements, and combine them into a new (virtual) product which can then be sold on the market. This business model already exists, but is supported completely manually. Since many ISPs can provide the basic services at different levels and with various pricing models, dishonest providers could claim higher QoS level to attract interested parties. The standard way to prevent this is to allow users to evaluate a service and provide feedbacks. However, the feedback mechanism has to ensure that false ratings, for example, badmouthing about a competitor's service or pushing one's own rating level by fake reports or collusion with other malicious parties, can be detected and dealt with. Consequently, a good service discovery engine would have to take into account not only the functional suitability of services but also their prospective quality offered to end-users by assessing the trustworthiness of both providers and consumer reports. According to several empirical studies (Despotovic & Aberer, 2004a; Jøsang *et al.*, 2007), the issue of evaluating the credibility of user reports is one of the essential problems to be solved for online service provisioning.

The QoS-based service selection approach is developed under two basic assumptions which are reasonable and realistic in e-Business settings: First, we assume *probabilistic behavior of services and users*. This implies that the differences between the real quality conformance which users obtained and the QoS values they report follow certain probability distributions. These differences vary depending on whether users are honest or cheating as well as on the level of changes in their behaviors. Secondly, we presume that *there exist a few trusted third parties*. These well-known trusted agents always produce reliable QoS reports and are used as trustworthy information sources to evaluate the behaviors of the other users. In reality, companies managing the service searching engines can deploy special applications themselves to obtain their own experience on QoS of some specific Web services. Alternatively, they can also hire third party companies to do these QoS monitoring tasks for them. For example, sites specializing in QoS monitoring (AlertSite.com, Dot-Com Monitor, Empirix) provide such a service. In contrast to other models (Maximilien & Singh, 2002; Ouzzani & Bouguettaya, 2004; Patel *et al.*, 2003; Ran, 2003; Tian *et al.*, 2003) we do not deploy these agents to collect performance data of all available services in the registry. Instead, we only use a small number of them to monitor QoS of some selected services because such special agents are usually costly to setup and maintain.

The QoS-based service selection and ranking algorithm we describe in this paper is a part of our overall distributed Semantic Web service discovery approach, which is published elsewhere (Vu *et al.*, 2005b, 2006, 2007). During the service discovery phase, after the functional matchmaking at a specific registry, a list of Web services with similar functionalities is obtained from the matchmaking of a service discovery framework, i.e., the services fulfilling all user's functional requirements. These services are then selected and ranked based on their predicted QoS values, taking into consideration the explicit quality requirements of users in the queries. The output of the selection and ranking algorithm is the list of Web services fulfilling all quality requirements of a user, ordered by their prospective levels of satisfaction of the given QoS criteria. So as to perform this selection and ranking accurately, the quality prediction is based on user reports on QoS of all services, which are collected periodically over time, on the quality levels promised by the providers, as well as the reliability of this information.

The main contribution of this work is a QoS-based Web service selection and ranking approach which is expected to be accurate, efficient and reliable. First, the issue of trust and reputation management has been taken into account adequately when predicting service performance and ranking services based on their past QoS data. Experimental results have shown that the newly proposed service selection and ranking algorithm yields very good results under various cheating behaviors of users, which is mainly due to the fact that *the use of trusted third parties observing a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments*. This is particularly important as without cheating detection, service providers would have much incentives to generate lots of false reports in order to obtain higher ranks in the searching results, thereby having higher probability to be selected by clients and gain more profits. Second, the proposed algorithm is semantic-enabled by offering support for the semantic similarity among QoS concepts advertised by providers and the ones required by users. This allows the QoS-based service selection process to work more flexibly and produce more accurate results.

The presentation of this work is organized as follows. First, I briefly mention the related work in section 5.1.2. Section 5.1.3 presents the general trust and reputation management model in a Web service discovery scenario. The proposed QoS-based service selection and ranking approach is described in detail in section 5.1.4. I will discuss various experimental results in section 5.1.5 and conclude the paper in section 5.1.6.

5.1.2 Related work

Although the traditional UDDI standard, at the time of this work is version 3.0.2[1], does not refer to QoS for Web services, many proposals have been devised to extend the original model and describe Web services' quality capabilities, e.g., QML, WSLA and WSOL (Dobson, 2004). The issue of trust and reputation management in Internet-based applications has also been a well-studied problem (Despotovic & Aberer, 2004a; Jøsang *et al.*, 2007).

(Chen *et al.*, 2003) proposes the UX architecture that uses dedicated servers to collect feedback of consumers and then predict the future performance of published services. (Bilgin & Singh, 2004) proposes an extended implementation of the UDDI standard to store QoS data submitted by either service providers or consumers and suggests a special query language (SWSQL) to manipulate, publish, rate and select these data from repository. Kalepu et al (Kalepu *et al.*,

[1] http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

2004) evaluate the reputation of a service as a function of three factors: ratings made by users, service quality compliance, and its verity, i.e., the changes of service quality conformance over time. However, these solutions do not take into account the trustworthiness of QoS reports produced by users, which is important to assure the accuracy of the QoS-based Web service selection and ranking results. (Liu *et al.*, 2004) rates services computationally in terms of their quality performance from QoS information provided by monitoring services and users. The authors also employ a simple approach of reputation management by identifying every requester to avoid report flooding. In (Emekci *et al.*, 2004), services are allowed to vote for quality and trustworthiness of each other and the service discovery engine utilizes the concept of distinct sum count in sketch theory to compute the QoS reputation for every service. However, these reputation management techniques are still simple and not robust against various cheating behaviors, e.g., collusion among providers and reporters with varying actions over time. Consequently, the quality of the searching results of those discovery systems will not be assured if there are lots of colluding, dishonest users trying to boost the quality of their own services and badmouth about the other ones. (Day & Deters, 2004) suggests augmenting service clients with QoS monitoring, analysis, and selection capabilities, which requires each service consumer would have to take the heavy processing role of a discovery and reputation system. Other solutions (Maximilien & Singh, 2002; Ouzzani & Bouguettaya, 2004; Patel *et al.*, 2003; Ran, 2003; Tian *et al.*, 2003) use third-party service brokers or specialized monitoring agents to collect performance of all available services in registries, which would be expensive in reality. Though (Maximilien & Singh, 2002) also raises the issue of accountability of these service brokers in their system, the evaluation of trust and reputation of these agents is still an open problem.

The presented QoS provisioning model is grounded on previous work of (Chen *et al.*, 2003; Kalepu *et al.*, 2004; Liu *et al.*, 2004; Ran, 2003; Tian *et al.*, 2003). The trust and reputation management of this work is most similar to that of (Guha *et al.*, 2004; Richardson *et al.*, 2003), yet the idea of distrust propagation is exploited more concretely by observing that trust information from a user report can also be used to reveal dishonesty of other reporters and by allowing this distrust to be propagated to similar ones. Other ideas of the trust and reputation management method are based on (Aberer & Despotovic, 2001; Cornelli *et al.*, 2002; Whitby *et al.*, 2005).

5.1.3 A reputation-based service quality management approach

Fig. 5.1 depicts a systematic view of the Web service management life-cycle with respect to quality evaluation. Service providers publish advertisements of their services with promised QoS levels on a variety of publishing media. Popular media are public and corporate (private) UDDI business registries, professional forums and social networks, or traditional Web portals. Potential service consumers typically use a search engine to discover functionally-matched services with a desired QoS level and select one among them to invoke. In practice, such a search engine in Figure 1 may represent either a dedicated Web service search engine, e.g., Seekda[1]

[1] http://seekda.com/

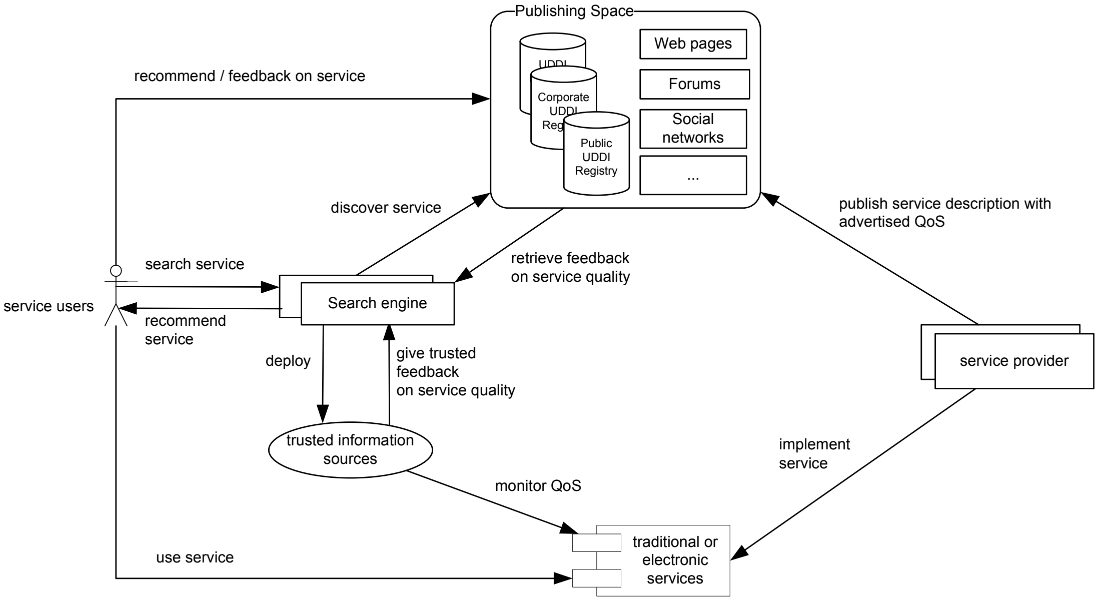


Figure 5.1: An overview of the reputation-based approach to Web service quality management.

or a manual search process performed by the user, possibly with traditional Web search tools. A search engine collects feedback on different QoS properties of available Web services from various information sources, including trusted monitoring parties. It then estimates the QoS of available services, performs the ranking and recommends the selection of the best services according to user's QoS criteria.

The interaction among participants in a QoS-based service selection and ranking system is represented in Fig. 5.2 where S_0,..,S_n are Web services, U_0,...,U_m are service users, RP_0,...,RP_k are registries from which quality advertisements on different services are collected. T_0,...,T_r are the trusted QoS monitoring agents providing reliable QoS ratings (possibly at certain cost) that will be used as referenced sources for the trust evaluation approach later on.

A user (U_j) who observes a normalized QoS conformance value x_{ij} (see Definition 11 in the following section) from a service S_i may report a value y_{ij} to a registry RP_k, which manages the quality description of S_i as advertised by its provider. This registry will collect users' quality feedbacks on all of its managed Web services to predict their future performance and support the QoS-enabled service discovery based on these data. Note that a user generally reports a vector of values representing its perception of various quality parameters from a service. Also, x_{ij}s and y_{ij}s are QoS conformance values, which already take into account the quality values advertised by providers (see Definition 11 in the next section). In this work, only the selection and ranking of services with reputation management in one registry peer is considered. The study of interaction among different registries is subject to future work.

Given the above interaction model, a number of observations can be made. An honest user

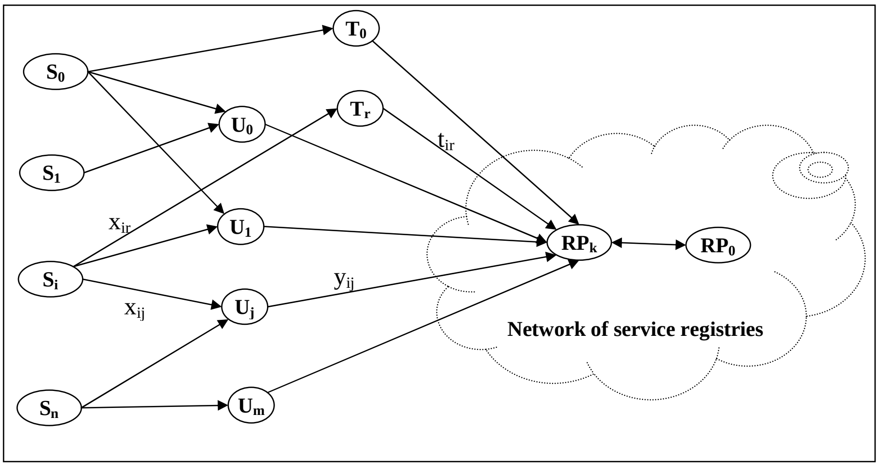


Figure 5.2: The system model of a QoS-based service discovery scenario

will report $y_{ij} = x_{ij}$ in most cases (in reality, they may not report if not willing to do so). On the other hand, a dishonest reporter who wants to boost the performance of S_i will submit a value $y_{ij} > x_{ij}$. Similarly, cheating reports generated by S_i's competitors will report $y_{ij} < x_{ij}$ to lower its QoS reputation. In addition, (colluding) liars may sometimes provide honest reports and behave dishonestly in other cases (Despotovic & Aberer, 2004a). Accordingly, we presume that the differences between y_{ij}s and x_{ij}s follow certain distribution types whose expected values depend on the behavior of the user, i.e., approximate to 0 for honest users and different from 0 in the case of liars. We use this assumption since in general dishonest users are likely to change their actions over time in order to hide their cheating behaviors with *occasionally* reliable reports. Moreover, as the quality parameter values of a service really depend on many environmental factors, even trusted agents and honest users may obtain and report those values a little different to each other when talking about the same service. However, the expected value of trustworthy reports on QoS of S_i, e.g., the average of corresponding reliable y_{ij} values, will reflect its real quality capability. The expected values of these distributions, e.g., means, normally represent the behaviors of users, whereas other parameters, e.g., standard deviations, will represent the uncertainty in actions of users, either accidentally or by purpose. My goal is not only to evaluate whether a user is honest but also to compute the expected conformance values from the reports of the most honest users, e.g., the average of values y_{ij}s by the most reliable reporters, from which the future quality of S_is will be predicted.

5.1.4 QoS-based service selection and ranking

In this approach, quality properties of (Semantic Web) services are described by concepts from a QoS ontology and then embedded into service description files using techniques suggested by WS-QoS (Tian *et al.*, 2003) and Ran (Ran, 2003). The value of a quality parameter of a Web service is supposed to be *normalized* to a non-negative real-valued number regarding service-specific and call-specific context information where *higher normalized values represent higher levels of service performance*. It is noteworthy that in practice, the issue of normalizing the values of various quality attributes is complicated and deserves another work of its own. However, with most frequently used QoS concepts, e.g., reliability, execution-time, response-time, availability, etc., the answer is well-defined and straight-forward. For instance, a Web service with a normalized QoS parameter value for reliability of 0.90 will be considered as more reliable to another one with a normalized reliability value of 0.50. In this case the normalized reliability is measured as its degree of being capable of maintaining the service and service quality over a time period T. The ontology language used to describe service semantics and to define the QoS ontology is WSMO[1], but other models, e.g., OWL-S[2], would also be applicable. For experimental evaluations, a simple QoS ontology has been developed for the VISP use case including the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, etc. Users and providers are assumed to share a common ontology to describe various QoS concepts. This QoS-enabled Semantic Web service discovery framework is described in detail elsewhere (Vu *et al.*, 2005b, 2006, 2007).

5.1.4.1 Predicting service performance

In order to predict the quality of a service S_i, QoS feedbacks on its performance over a time period W are collected and a linear regression approach is used to predict its future quality conformance from past data. First let us begin we two basic concepts.

Definition 11 The quality conformance value c_{ij}^k of a service S_i in providing a quality attribute q_{ij} at time k is defined as $c_{ij}^k = \frac{d_{ij}^k - p_{ij}^k}{p_{ij}^k}$ where d_{ij}^k is the normalized value of q_{ij} that S_i actually delivers to a specific user at time k and p_{ij}^k is the corresponding normalized QoS value promised by S_i's provider at k.

Definition 12 A user QoS report R, either by a normal service consumer or by a trusted monitoring agent, is a vector $\{u, S_i, t, L\}$, where u is the identifier of the user that produced this report, S_i is the corresponding service, t is the timestamp of the report and L is a quality conformance vector of $\{q_{ij}, c_{ij}^t\}$ pair values, with q_{ij} being a QoS attribute offered by S_i and c_{ij}^t being q_{ij}'s quality conformance that S_i provides to this user at time t.

Note that a normalized QoS conformance value already considers the discrepancies between quality level advertised by the provider and the actual quality experienced by the use, the quality prediction in this work already covers possibly cheating behaviors of providers. In order to filter out as much dishonest reports as possible and to take only the most reliable ones in the QoS prediction, a reputation-based evaluation of trustworthiness of different reports is applied via two steps: a report preprocessing and a report clustering phase. The first phase evaluates the reliability of collected reports by applying a trust-and-distrust propagation approach, which

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

[2] http://www.daml.org/services/owl-s/

relies on some initial trusted reference reports produced by special monitoring agents. Two QoS reports are considered as *comparable* if they are related to the same service during a specific time interval δ_t and as *incomparable* otherwise. Generally, this δ_t can be set as big as the length of the period during which the corresponding service provider does not change the promised quality values of this service. Two *comparable* QoS reports are considered to be *similar* if the squared Euclidean distance between their conformance vectors is less than a specific threshold. On the contrary, they are regarded as *dissimilar* if this distance is greater than another threshold value.

The report preprocessing step follows a trust-distrust propagation approach as shown in Algorithm 5. n_{ch} and n_{h1}, n_{h2} ($n_{h1} < n_{h2}$) are threshold values to estimate a user as cheating or honest regarding to the similarity of its reports to other cheating/honest ones (line $9, 17$ and 18). N and T represent for how long and how frequent a user stay in and submit QoS reports to the system. The values of $n_{h1}, n_{h2}, n_{ch}, N$ and T are design parameters to be chosen depending on properties of the collected reports after running the algorithm several times. Generally, higher values of these parameters stand for higher level of caution when estimating behaviors of users regarding current evidences against them. Note that the algorithm always terminates since the number of users is finite and since after each iteration of this algorithm (line 11 or 19), either the behavior of one more user (as cheating or honest) is revealed or the loop terminates.

Algorithm 5 QosReportsPreprocessing()

```
 1: all trusted agents are marked as honest users;
 2: all reports of trusted agents are marked honest;
 3: repeat
 4:   all unmarked reports of each cheating user are marked cheating;
 5:   for each unmarked report do
 6:     if this report is dissimilar from an honest report then mark it cheating;
 7:     if this report is similar with a cheating report then mark it cheating;
 8:   end for
 9:   users with at least $n_{ch}$ reports similar with cheating ones are marked cheating;
10:   users with at least $N$ reports in at least $T$ different time are marked frequent;
11: until there is no new cheating user discovered;
12: repeat
13:   all unmarked reports of each honest user are marked as honest;
14:   for each unmarked report and each report marked cheating do
15:     if this report is similar with an honest report then mark it honest;
16:   end for
17:   unmarked users with at least $n_{h1}$ reports similar with honest ones are marked honest;
18:   users marked as cheating and having at least $n_{h2}$ reports similar with honest ones are re-marked honest;
19: until there is no new honest user discovered;
```

After the preprocessing phase finishes, a certain number of cheaters and honest users are identified. However, this trust-distrust propagation phase may not be able to evaluate the reliability of all reports, i.e., certain users may use only some services that are not popular to other users, and as if the values of n_{h1}, n_{h2} and n_{ch} are set too high in Algorithm 5.

Therefore, in the next step we have to estimate the trustworthiness of the remaining reports whose reliability has not been evaluated. To achieve this, it is assumed that collusively cheating users will cooperate with each other in order to influence the system with their dishonest feedbacks. As a result, users within each group will produce similar values and naturally form different clusters of reports. Thus it is possible to apply well-known data-mining techniques in this situation to discover various existing clusters of reports related to those user groups. In this work, a standard *convex k-mean clustering algorithm* is used on each set of QoS reports related to a service during the time interval δ_t with the following metrics: the distance between two *comparable* reports is defined as the Euclidean squared distance between two corresponding quality conformance vectors and the distance between two *incomparable* ones is assigned a large enough value so that these reports will belong to different clusters.

After the trust and reputation evaluation in the two above phases, for each service S_i, we will have a list G_i of groups of reports on QoS of S_i over time. Generally, G_i includes the groups containing those reports that were previously marked as honest/cheating during the trust-distrust propagation phase, as well as other clusters of reports obtained after the clustering step. We will assign the credibility w_i^g of a report group $g_i \in G_i$ as follows. Firstly, all dishonest ratings are eliminated: we would assign $w_i^g = 0.0$ for those report groups marked as cheating during the trust-distrust propagation. If there exists the group g_i^0 of reports previously marked honest during that step, we assign $w_i^{g_0} = 1.0$ whereas letting $w_i^g = 0.0$ for the remaining groups. Otherwise, w_i^g is estimated proportional to the probability that the group g_i is trustworthy among all the other groups. The used heuristic is to assign higher weight to clusters that are more densely populated, having bigger size and with more *frequent* users. This assumption is reasonable, as the reports of independent cheaters are likely to be scattered. In the case liars cooperate with each other to cheat the system, there would be strong correlation among reports of such colluding cheaters. The size of their corresponding clusters will not exceed those consisting only of honest reports as it would be too costly to dominate the system with numerous and clustered dishonest ratings. Even if dishonest providers try to produce lots of more well-similar reports so that they could get high influences to the final ranking results, these values will be separated from honestly reported values and therefore are likely to be discovered during the distrust-propagation phase (line 3 to line 11 in Algorithm 5), provided there are sufficient trusted reports to use as reference.

Specifically, w_i^g could be estimated based on the following information: the number of users in the cluster g_i ($size_i^g$), the number of users producing reports in all clusters of G_i ($allusers_i$), the number of frequent users in this cluster ($frequent_i^g$), the total number of frequent users in all clusters of G_i ($allfrequent_i$), as well as the average distance d_i^g from the member reports of cluster g_i to its centroid values. Based on the model in section 5.1.3, the reliability of a report would depend on the distance between its conformance vector and that of an honest report. Therefore, the similarity among reliability of different reports in one cluster g_i would be inversely proportional to its d_i^g value. Furthermore, a report in g_i would be honest in two cases: (1) it is submitted by a *frequent and honest* user; (2) it is produced by an *infrequent and honest* user. Let P_{freq} [and P_{infreq}] be the probability that this report is of a frequent user [and an infrequent] user, respectively, and let P_{freqr} [resp. $P_{infreqr}$] be the probability that frequent [resp. infrequent] users report reliably, then we have $w_i^g = \frac{C}{d_i^g}.(P_{freq}.P_{freqr} + P_{infreq}.P_{infreqr})$, where $P_{freq} = \frac{freq_i^g}{size_i^g}$ and $P_{infreq} = 1 - P_{freq}$. P_{freqcr} and $P_{infreqcr}$ could be estimated by comparing reports of trusted agents with those of sample frequent/infrequent users. The value of C represents the belief in the relation between the density of a report cluster and the reliability of its member reports, which is considered as a system parameter to be set by experience.

The future quality conformance $\hat{C}_{ij}$ of a service S_i in providing a QoS attribute q_{ij} is predicted using linear regression. Specifically, we have: $\hat{C}_{ij} = LinearRegression(\overline{C}_{ij}^t)$, $t \in \{0, 1, \ldots, (W-1)\}$, where $\overline{C}_{ij}^t$ is the evaluated QoS conformance value of the quality parameter q_{ij} at time t, and W is the window size of the service historical performance.

We compute $\overline{C}_{ij}^t$ as the average of conformance values reported by various user groups in the system at that specific time point, using the evaluated credibility w_i^gs as weights in the computation. In other word, $\overline{C}_{ij}^t = \frac{\sum_{g_i \in G_i} w_i^g C_{ij}^t}{\sum_{g_i \in G_i} w_i^g}$ where C_{ij}^t is the mean of conformances of a report group g_i on S_i's quality attribute q_{ij} at time t, i.e., a centroid value of a cluster/group of reports produced after the trust and reputation evaluation phase. Given the assumption on probabilistic behavior of users and services as in section 5.1.3, $\hat{C}_{ij}$ is seen as an estimate of the expected value of S_i's QoS conformance in providing quality attribute q_{ij} to users.

5.1.4.2 QoS-based service selection and ranking

The QoS requirements in a user query are definedas a vector Q of triples $\{q_j, n_j, v_j\}$ where q_j represents for the required QoS attribute, n_j is the level of importance of this quality attribute to the user and v_j is the minimal delivered QoS value that this user requires. To rank services according to its prospective level of satisfying user's QoS requirements, the Simple Additive Weighting method is used. This method, despite its simplicity, has been empirically shown to produces ranking results very close to those of more sophisticated decision making techniques (Ouzzani & Bouguettaya, 2004).

Thus, the QoS rank of a service S_i in fulfilling all quality criteria depends on the weighted sum $T_i = \frac{\sum_{q_j \in Q} n_j.P_{ij}}{\sum_{q_j \in Q} n_j}$ in which $P_{ij} = w_{ij}.\widehat{nd}_{ij}$ represents the capability of S_i in providing the quality parameter q_{ij} for users at the query time. The value $\widehat{nd}_{ij} = \frac{\hat{d}_{ij} - v_j}{v_j}$ evaluates the difference between the QoS value $\hat{d}_{ij}$ of the quality attribute q_{ij} that S_i is able to offer to its users according to the prediction and the quality v_j of q_j required by the user. From Definition 11, we can compute $\widehat{nd}_{ij} = \frac{(1+p_{ij}).\hat{C}_{ij} - v_j}{v_j}$, where $\hat{C}_{ij}$ is the predicted QoS conformance value for quality attribute q_{ij} and p_{ij} is the corresponding QoS value promised by provider of S_i at current time. w_{ij} is a weight proportional to the semantic similarity m_{ij} between q_{ij} and the QoS ontology concept q_j required by the user, i.e., the degree of match as in (Paolucci *et al.*, 2002). In other words, higher ranks are given for services offering the most similar QoS concepts at the higher levels compared to the ones required by users. In our program we simply use the following definition to evaluate w_{ij}:

$$w_{ij} = \begin{cases} 1.0 & \text{if } m_{ij}{=}\mathit{exact}; \text{(i.e., } q_{ij} \text{ is equivalent to } q_j) \\ 0.5 & \text{if } m_{ij}{=}\mathit{pluggin}; \text{(i.e., } q_{ij} \text{ is more general than } q_j) \\ 0.0 & \text{if } m_{ij} \in \{\mathit{subsume}, \mathit{failed}\}; \text{(i.e., otherwise)} \end{cases} \tag{5.1}$$

In order to avoid the time-consuming semantic reasoning step and to accelerate the selection of only services fulfilling all required QoS parameters, an inverted *QoS matching table* is used to store the matching information for frequently accessed quality attributes, similar to the idea proposed by Srinivasan et al (Srinivasan *et al.*, 2004). With each attribute q_j in this table, a list $Lqos_j$ of records $\{S_{ij}, w_{ij}, \hat{d}_{ij}\}$ is computed where w_{ij}, $\hat{d}_{ij}$ are estimated as above and S_{ij} identifies a service having certain support for q_j. Given the list L of services with similar functionalities, the discovery engine performs the QoS-based service selection and ranking process as in Algorithm 6.

Algorithm 6 QosSelectionRanking(ServiceList L, ServiceQuery Q)

```
Derive the list of QoS requirements in Q: $L_q = \{[q_1, n_1, v_1], ..., [q_s, n_s, v_s]\}$
Initialize $Score[S_{ij}] = 0.0$ for all services $S_{ij} \in L$;
for each quality concept $q_j \in L_q$ do
    for each service $S_{ij} \in L$ do
        Search the list $L_{qos}$ of $q_j$ for $S_{ij}$;
        if $S_{ij}$ is found then
            $Score[S_{ij}] = Score[S_{ij}] + \frac{n_j . w_{ij}}{\sum n_j}(\frac{\hat{d}_{ij} - v_j}{v_j})$;
        else
            Remove $S_{ij}$ from $L$;
        end if
    end for
end for
Return the list $L$ sorted in descending order by $Score[S_{ij}]$ s;
```

5.1.5 Experimental results and discussion

The presented service selection and ranking algorithms are implemented as a QoS evaluation module in a Semantic Web service discovery component (Vu *et al.*, 2006). The effectiveness of these algorithms are then studied under various settings. Three representative quality parameters are chosen, namely *availability*, *reliability* and *execution-time* taken from the VISP case study.

I then observed the dependency between the quality of selection and ranking results and other factors, such as the percentage of trusted users and reports, the rate of cheating users in the user society and the various behaviors of users. Specifically, the effectiveness of the service discovery is measured by estimating differences in the quality of results when running evaluations in four different settings: In the *ideal* case the discovery engine has complete knowledge, such that it knows correct QoS conformance values of all published services in the system over a time window W and performs the selection and ranking of services from this ideal data set.

In the *realistic* case, the presented algorithms with the proposed approach of trust propagation and report clustering to filter out biased reports are run to evaluate the reliability of every report. The *optimistic* case corresponds to the selection and ranking services without considering any trust and reputation issues, i.e., the system simply uses the average of all reported conformance values to predict services' performance and to perform the QoS ranking. The *naive* case corresponds to the selection of services based only on quality values promised by the providers, i.e., the system trusts all providers completely. The goal of experiments in this section is to show that the obtained results of the QoS-based service discovery process are more accurate and reliable in the realistic case with various cheating behaviors of users, they would be much worse in the optimistic case, and the worst with the naive method thus clearly showing the contribution of my approach.

The quality of selection results produced by a selection and ranking algorithm can be measured by four parameters, namely *recall*, *precision*, *R-precision* and *Top-K precision*, of which the *R-precision* is the most important quality parameter as recognized by the Information Retrieval community. Recall that *R-precision* is the fraction of truly relevant results in the R highest ranked results, where R is the number of truly relevant results obtained with a perfect service search engine. In this work, these parameters generally represent the fraction of services that are most relevant to a user among all returned services in terms of their *real* QoS capabilities. Apparently, the results would be the best in the ideal case with a perfect search engine, i.e., its recall, precision, R-precision and Top-K precision parameters are all equal to 1.0. Therefore, the results of the ideal case are used as a reference to compare with the quality parameters in the other three situations. Due to space limitations only the *R-precision* values are shown as the most representative experimental results in this section. The other performance measures and the weighting Spearman's correlation between the ideal ranking result and the ranking vector returned by my method are also measured. Similar performance trends were observed.

The experiments were prepared with the probabilistic assumption on the behavior of service providers and consumers. In this work I only present the experiments with Gaussian (normal) distributions. The extension to other probabilistic distributions is subject to future work. The community of service consumers was modeled as a collection of different types of users. As mentioned in section 5.1.3, honest users and trusted agents would report values with the difference $D_h \sim Normal(0, {\sigma_h}^2)$ to the real QoS conformance capabilities of services. On the contrary, cheaters would report values with the difference $D_c \sim Normal(M_c, {\sigma_c}^2)$ to the real quality conformances that they had obtained. The values of the mean M_c varied according to the purpose of the cheaters, i.e., to advertise or to badmouth a service. The values of σ_c represented the variation in reported values of users among different quality attributes of different services. Users with higher values of σ_c had higher levels of inconsistency in their behaviors and therefore were harder to be detected. These liars are then further divided into three sub-types: *badmouthing users* who mostly reported badly about services of their competitors, *advertising users* who usually exaggerated performance of their own services and *uncertain users* with indeterministic actions and who might act as advertising, badmouthing or even as honest users.

The values of M_c for each type of cheaters were set in the most pessimistic situation, making the testing environment highly hostile. Specifically, cheaters had their corresponding M_cs set to high/low enough values such that badmouthing users would succeed in pushing services of their competitors out of the ranking results and advertising users would be able to raise the QoS ranks of their own services, provided that their reports had been taken into the predicting process of the QoS-based service discovery engine. This setting is realistic because in business, companies generally have knowledge of the base requirements of their users as wells as owning certain statistics of their competitors' capabilities. More complicatedly, uncertain users had their M_c values belonging to N_c specific values each of which was a randomized real value, with N_c was the number of groups of cheaters with varied behaviors. These types of liars would be harder to be detected since their M_c values were uniformly distributed around 0.0. Though they did not contribute directly to the boosting and lowering the reputation of certain services, their reports were not so dissimilar from honest reports in most cases and therefore they would act as good recommenders for other badmouthing/advertising guys. The scalability of the approach was also tested by increasing number of services, users and QoS reports while keeping the percentage of user types and other parameters the same. The remaining experiments were run with 1000 users who produced a total of 50000 QoS reports on 200 services during a time window of length $W = 5$. The results of each experiment were averaged over 10 runs.

As a first question, the effects of the trusted reports on the quality of results in the realistic case are studied. Specifically, I wanted to observe the effects of the percentage of the services monitored by trusted agents $F_{special}$ to the results of the QoS-based service selection and ranking algorithm expressed by *R-Precision* values. I increased $F_{special}$ from 1% to 10%, letting the percentage of trusted users/reports increase from 0.1% to 1% with the increment of 0.1% each step. The results of this experiment are shown in Fig. 5.3.

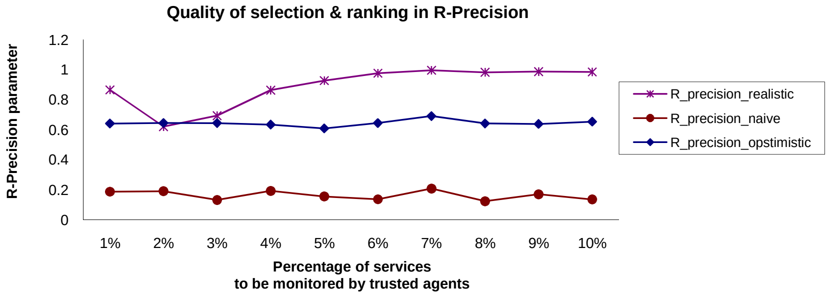


Figure 5.3: $F_{special}$ vs. *R-Precision*

Correspondingly, Fig. 5.4 shows the percentage of cheating and honest reports correctly identified during the report preprocessing phase with the trust-distrust propagation method.

In this experiment, a very high number of cheaters are assumed (74% of the total users)

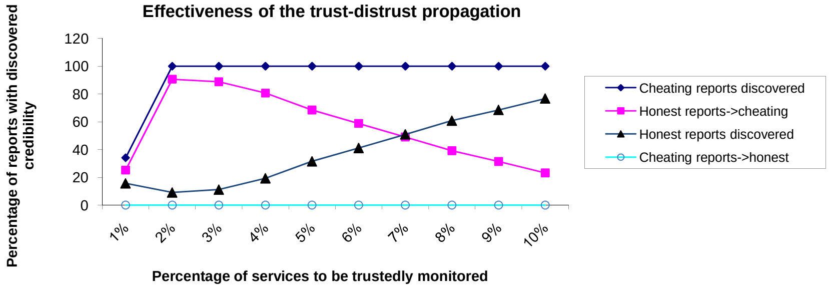


Figure 5.4: $F_{special}$ vs. *Effectiveness of the trust-distrust propagation.*

consisting of badmouthing users, advertisers and five different groups of uncertain users. Various experiments were also carried out to study the effects of σ_c's and σ_h's values: it turns out that the trust-based selection and ranking algorithm performed very well with different settings of σ_c provided the ratings of the honest user population had small uncertainty, i.e., $\sigma_h < 0.1$. When the reports from trusted and honest users have higher uncertainty the misclassification errors by using the trust and distrust propagation became higher and thus the quality of the service selection and rank in terms of *R-precision* deteriorated. However even in such worst cases the results are still comparable with those of the optimistic approach and still much better than those of the naive approach. The variances of values in cheating reports did not have clear influence on the result, and the proposed trust management approach is good against those cheating users whose reports have different values with high uncertainty. This effect is intuitive, since my proposed approach relies on correlation among ratings from trusted and honest users to discover reliable ratings and isolate unreliable ones. Given the fact that the QoS conformance values in the simulation are in the interval $[-1, 1]$, the standard deviations of cheating reports are kept very high ($\sigma_c = 0.5$) and those of trusted and honest users are kept at an acceptably low level ($\sigma_h = 0.01$). With the increase of $F_{special}$, almost all cheating reports and an increasing percentage of honest reports can be correctly identified. Accordingly, the quality of the service selection and ranking results was significantly increased as well. The clustering phase was actually not performed with $F_{special} > 1\%$ because above this threshold, using only the trust-distrust propagation was enough to evaluate the credibility of all reports. Although a lot of honest reports were wrongly identified as cheating, which was due to the cautious approach in estimating report reliability, the quality of the results was always very good if $F_{special}$ was kept high enough (about 5%). The results were the worst with $F_{special}$ around 2% and 3%. In these cases, as the trust-propagation was not effective and did not identify enough honest reports, and the quality prediction had to (partly) use the quality values advertised by providers instead. When $F_{special}$ was very small (1%), the result was still acceptable (*R-Precision* $= 0.8$), since in this situation the algorithms actually still benefit from the use of the report clustering phase. i.e., there were enough reports with unknown reliability after the preprocessing of reports such that the clustering step had enough input data to process. Given this assumption, to maximize the effectiveness of the proposed trust management approach, it is recommended that we either use only the clustering phase (if $F_{special}$ is small), or if we want to use the trust-distrust propagation, $F_{special}$ should be sufficiently high ($F_{special} \approx 3 - 5\%$). As a whole, the result of the proposed selection and ranking algorithm with trust and reputation management in place is much better than that of the optimistic and the naive cases, as expected.

The next question to be investigated is the effects of the fraction of cheaters to the quality of results. I gradually increased the total percentage of all types of cheating users ($F_{cheating}$), which consists of badmouthing, advertising and uncertain users, from as small as 4% to as much large as 94%p. More specifically, the percentage of badmouthing and advertising users/reports are raised from 3.33% to 78.33%. Malicious reports from these dishonest users form five different groups of uncertain users with corresponding percentage of dishonest reports increased from 0.67% to 15.67%. This setting represents the realistic case when there are various types of dishonest providers colluding with the generated cheating users to boost the reputation of their own services and badmouth other ones, which could be considered as the most important case when there are various types of users with changing behaviors. The results for this experiment are shown in Fig. 5.5. We kept the percentage of trusted reports at 0.5% and let $F_{special} = 5.0\%$, as an acceptable fraction collected from observations in the first experiment. The standard deviations of of cheating and honest reports were kept at $\sigma_c = 0.5$ and $\sigma_h = 0.01$ respectively.

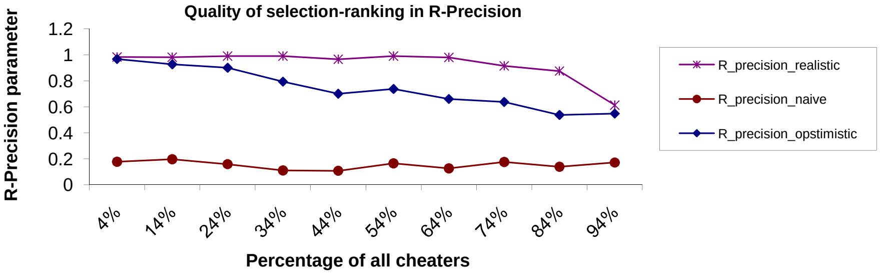


Figure 5.5: $F_{cheating}$ vs. *R-Precision*

With the reduction of honest users and the corresponding increase of $F_{cheating}$, *R-Precision* was also reduced. However, the quality of the results in the realistic case was always much better than that of the optimistic case and the naive case. Even when the fraction of cheaters $F_{cheating}$ was very high (0.84), the *R-Precision* parameter value in the realistic case was still acceptable (higher than 0.8). On the other hand, the quality of results without taking into account trust and reputation management issues, i.e., the optimistic and the naive case, dropped dramatically in hostile settings. This phenomenon was due to the fact that in a user community with a very high cheating rate, the proposed trust management mechanism could help discover and filter out almost all unreliable reports, as shown in Fig. 5.6.

From these experiments a number of conclusions can be drawn. Regarding its efficiency and

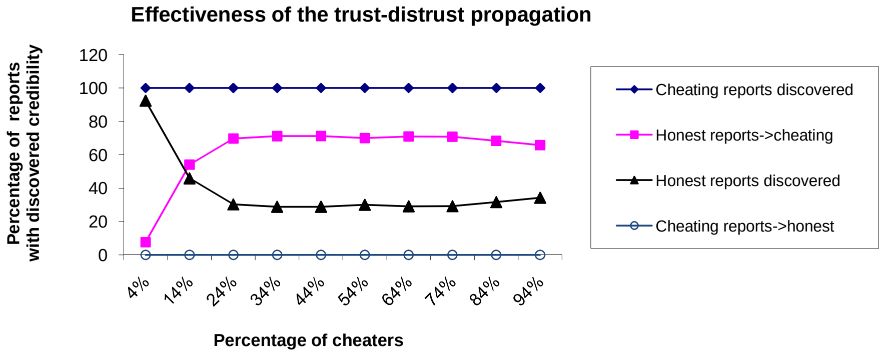


Figure 5.6: $F_{cheating}$ vs. *Effectiveness of the trust-distrust propagation.*

effectiveness, as the trust and reputation evaluation phase uses a trust-distrust propagation and a data-mining (report clustering) methods, the computational cost can be relatively high. Fortunately, in the current application scenario, the necessary computations involve mainly local processing and thus does not require much communication overheads. Additionally, they could be done *off-line* on a periodical basis and therefore will not affect much to the system performance. Another important observation is that almost no cheating report was wrongly evaluated as honest even in very hostile settings due to the cautious reputation evaluation mechanism. Therefore, in reality one can observe the results of the trust-distrust propagation step and incrementally adjust the parameters of the system, e.g., increase $F_{special}$, in order to collect enough honest reports for the performance prediction phase. We observe that *the use of trusted third parties monitoring a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments.* However, this effectiveness mainly depends on the following properties of the set of service users and their corresponding reports:

1. *The overlaps in the set of users for each service*, i.e., whether the services monitored by trusted agents have many users who are also consumers of other services.

2. *The inconsistency in the behavior of users*, i.e., whether a user is honest while reporting on a service but behaves dishonestly on other cases.

3. *The high level of certainty in reports of trusted and honest users*, since our approach relies on the correlation among ratings of honest users to identify reliable reports and eliminate potentially biased ones.

These factors suggest that *trusted agents should be used to monitor the QoS of the most important and most widely-used services* in order to build a good reference data set for the estimation the behaviors of other users. Currently as the user reports are distributed uniformly for services in all experiments, this factor has not been taken into account. Additionally, other techniques can also be utilized to make this method more robust. For example, it is possible to *pre-select the important services to monitor* and *increase the number of them* as well as *keep the* *identities of those specially chosen services secret* and *change them periodically*. Thus, cheaters are not aware of which services they should report honestly in order to become highly-reputable raters. Such cheaters have to pay a very high cost to have great influences over other raters in the system. In reality, this also help us to reduce the cost of setting-up and maintaining trusted agents as we only need to deploy them to monitor changing sets of services at certain time periods.

5.1.6 Conclusion

This first part of the current chapter presents an application of reputation-based trust management techniques in a QoS-based service selection and ranking scenario. A novel reputation-based trust management approach to evaluate reliability of reports from different users has been proposed. It has been shown that such a trust management mechanism may help the selection and ranking of services to produce very good results in most cases. As the proposed reputation management mechanism is robust against various cheating behaviors, the results are generally of good quality even in hostile situations in which many different types of cheaters make up a high percentage of the overall users and report values with remarkable variances. By combining a trust-distrust propagation approach with a clustering method, a majority of cheaters and honest reports can be correctly identified to be used in the quality prediction. However, there are a lots of open issues and improvements that need further investigation in the current model. First of all, the selection of services to be monitored by trusted agents is an interesting point not yet to be mentioned. Another open problem is how to accurately predict the performance of newly published services with only few QoS reports and how to motivate users to evaluate services' performance and submit their feedback to the service search engine. The selection of an optimal configuration for many design parameters of the proposed solution is also an important question that needs to be investigated.

5.2 Secret Sharing in Distributed Online Social Networks

The second part of this chapter studies the use a computational trust model in a very different context. We will consider the application of trust management to enhance security of a threshold-based secret sharing protocol in a distributed online social network (DOSN), where users need a means to back up and recover their private keys in a network of untrusted servers. In this scenario, peers acting on behalf of users play the role of delegates that offer secret keeping services to each other. Selection of reliable delegates plays an important role in such a setting, as this directly affects the security of the system: delegates keeping shares of the secret may collude to steal the user's private keys.

I propose using two techniques to improve the system security: by selecting the most reliable delegates for keeping these shares and further by encrypting the shares with passwords. A mechanism is developed to select the most reliable delegates based on an effective trust measure. Specifically, relationships among the secret owner, delegate candidates and their related friends are used to estimate the trustworthiness of a delegate. This trust measure minimizes the likelihood of the secret being stolen by an adversary and is shown to be effective against various collusive attacks. Extensive simulations show that the proposed trust-based delegate selection performs very well in highly vulnerable environments where the adversary controls many nodes with different distributions and even with spreading of infections in the network. In fact, the number of keys lost is very low under extremely pessimistic assumptions on the adversary model.

5.2.1 Introduction

A threshold-based secret sharing scheme is a multi-party cryptographical protocol to enable a user to share her secret with only intended recipients in a distributed system (Schneier, 1995). A traditional (k, n)-threshold secret sharing protocol splits a secret into n parts (shares), any k of which (minimum) suffices to reconstruct the secret, e.g., the Shamir approach (Shamir, 1979). This approach can be well adapted to match the properties of peer-to-peer environments, and online social networking applications, where a user can not totally trust any other user, and no other single user is told the whole secret.

I propose a new application of a threshold-based secret-sharing protocol in a *distributed online social network* (DOSN). Such a system uses a distributed or a P2P infrastructure for its users' data management and storage, while providing functionalities of conventional (centralized) social networking sites such as *Facebook.com* or *Orkut.com*. There are various motivations for such a decentralized architecture, foremost among these being users' privacy and autonomy from not only fellow users but also from service providers. The vision of DOSN platforms has been presented in several recent works, e.g., (Buchegger & Datta, 2009; Buchegger *et al.*, 2009; Cutillo *et al.*, 2009; man Au Yeung *et al.*, 2009), or *Tribler.org*.

The application of threshold-based cryptographical protocols will be described through a concrete usage scenario. This scenario comes from the experience in the development of such a DOSN (Buchegger *et al.*, 2009), where users need a means to back up and recover their private keys in a network of untrusted servers[1]. This example will be also used to elaborate on the problem studied in this paper, as well as to define the scope of intended solutions for the problem. A practical realization of this scenario and its related solutions is the recovery of user's passwords in a distributed storage system such as *Wuala.com*.

Private key recovery example: Alice's computer crashed, so she must use another computer. She wants to log in to her online social network (a DOSN) from the new computer, retrieve associated data and resume her life online.

When Alice first created her account using the previous computer, the system generated a private key as a means of authentication associated with her username. The private key of Alice is the ultimate secret enabling her to manage her personal data, e.g., to edit a blog entry or to configure her privacy setting. In contrast to conventional web-based online social networks such as *Facebook.com*, where user data are stored at servers owned by the service providers, a DOSN platform enables Alice to store her personal data *mainly on her computer* to ensure her

[1] The two terms *secret* and *private key* will be used interchangeably henceforth

total control on these data. However, anticipating a future crash of the original computer and loss of the data stored locally on it, and also to increase data availability, Alice's data is also encrypted and replicated in other machines. As the private key is difficult to remember and can also be lost, it is also backed up: Alice split the key according to a (2,3)-threshold cryptography approach, and stored that in the network itself.

For enhanced security, each part of the key was encrypted by a *passphrase* chosen by Alice, resulting in $n = 3$ encrypted shares. Each of the three delegates Bob, Carol, and Dora is asked by Alice to keep a different encrypted share. These delegates are expected to only send a share to the user proven to be Alice.

Since the delegates may not know Alice a priori or they may not be able to meet in person, they need to verify Alice's identity (and thus ownership of the secret). Automatic verification, such as security questions/answers are applicable. These questions/answers can be possibly different for each delegate and digitally signed by Alice to prevent forgery. Upon successful verification of Alice's identity, a delegate sends back to Alice the locally stored encrypted share of Alice's secret. Alice recovers her private key by getting any two shares from three delegates, which she can decrypt using her private passphrase.

Unfortunately, there are several practical problems in using such a secret sharing scheme in a DOSN scenario.

- Users may be untrustworthy (*malicious*) when acting as delegates. An untrustworthy user may keep the shares to steal the secret for her own purposes, e.g., to control and steal Alice's private data. A user is untrustworthy either because she is curious or because her computer is controlled by a malicious software (an adversary). In the above example, the key of Alice is lost if (and only if) any $k \geq 2$ delegates among Bob, Carol, and Dora, are untrustworthy.
- The original secret can not be recovered without enough trustworthy delegates available, e.g., both Bob and Carol are on vacation and turn off their computers. Also, delegates may send invalid shares to reject the owner's requests, either intentionally or accidental due to software bugs, network errors, etc.
- The secret owner may forget the passphrase or answers to secret questions, and cannot recover the secret. The passphrase or these answers may also be lost (weak passwords) and thus an adversary can easily use this information to steal the secret (identity theft).

The main focus of this work is on the first issue, as in online social networks the collusion among malicious delegates is even more feasible and detrimental. An adversary can control a large number of malicious users appearing as legitimate to coordinate the attack and steal the user's key. Having the private key, the adversary may tweak security options on the victim's machine, enabling malicious applications to control and use that machine for further attacks. The situation is even worse as a user usually trusts her friends, unaware of whether their machines are already under control of an adversary. This viral infection may spread through social links rapidly, potentially leading to an epidemic that may, at worst case, makes the whole system eventually collapse.

Towards the above problem, most enhancements of threshold-based secret sharing schemes include the possibility to verify the validity of a share, to change the threshold dynamically, or to improve the computational and communicating efficiencies of the approach (Schneier, 1995). Current solutions to protect keys are to encrypt shares with passwords or to use verifiable credentials. The *resilience* of such protocols under *collusive attacks of malicious nodes* are not yet sufficiently studied. There has been *little study of how to select the most reliable delegates* under various adversary distributions in a large distributed network and understand the impact of such selection to the security level of the whole system. The main reason for such limitations is that threshold cryptography is mostly used on systems under control of a single centralized provider, which is different from the current application context.

To improve security of the secret sharing protocol in such distributed scenarios, this work proposes a mechanism to select the most trustworthy delegates. Delegate trustworthiness is estimated by exploiting relationships among the secret owner, delegate candidates, and their related friends. This trust measure minimizes the likelihood of the secret being stolen by an adversary and is shown to be effective against various collusive attacks. Extensive simulation shows that compared to other approaches, e.g., (Xu *et al.*, 2008), such a trust-based selection performs very well in a variety of scenarios with several nodes under control of the adversary, with different distributions of adversarial nodes, and even with the spreading of infection from malicious nodes.

To the best of my knowledge, this approach is among the first ones applying threshold cryptographical protocols to enable secure secret sharing on distributed social networks. The improved secret sharing scheme also has other practical applications, such as to enable delegated access control on other distributed systems. For instance, in a P2P-based content sharing system, a peer may rely on trustworthy delegates to distribute the data encryption key to those peers whose identities are unknown beforehand[1] yet proven to be from a subscribed reader group. With the given key, authorized readers can then decrypt any data replica by the original author even if the author is unavailable.

To reduce the work scope, I do not focus much on the second issue: the impacts of delegate availability to reconstruct the backed-up secrets. In fact, delegate unavailability is not a major problem in a key recovery scenario, as recovery is assumedly unfrequent. Thus, if there are not enough delegates available at the moment to rebuild the secret, the owner may simply wait.

The third issue is related to the user's security awareness, which is orthogonal to the current problems being studied. Nevertheless, with a threshold cryptographical approach, even if a user may choose weak passwords, an adversary must collect at least k shares and successfully decrypt them to steal the key. Therefore, using threshold cryptography for key backup is a generalized and more secured backup procedure compared to conventional approaches, e.g., to backup the whole key on a single server on the network.

[1] Otherwise, a traditional PKI-based approach can be used, e.g., by encrypting the key with the public key of the authorized readers.

5.2.2 System model

5.2.2.1 Notations

Denote as $\mathcal{U}$ the set of users and let $\mathcal{D}_f$ be the set of possible types of relationships among them, e.g., family, close friends, colleagues, or acquaintances. Define the mapping $f : \mathcal{U} \times \mathcal{U} \rightarrow \mathcal{D}_f$ as relationships among these users $\mathcal{U}$. A distributed online social networking platform is formally defined as follows.

Definition 13 *A* distributed online social network*, or* DOSN *for short, is a triple $\langle \mathcal{U}, f, \mathcal{RP} \rangle$, where $\mathcal{U}$ is the set of users or computers and f denotes the social relationships among them. The mapping $\mathcal{RP} : \mathcal{U} \rightarrow 2^{\mathcal{U}}$ is the data replication strategy that defines the set of nodes in the system to store the data $\mathcal{R}_u$ of a user $u \in \mathcal{U}$.*

Def. 13 associates each user with a node, e.g., her main working computer, in the underlying distributed storage system. This assumption simplifies the problem and subsequent analysis while still being realistic as most users primarily use one computer to work. Henceforth, the notations peer, node, or user will be used interchangeably. It is the focus of this work to build a mechanism to share a secret key in a DOSN platform, with any replication strategy, securely and effectively under various adversarial attacks.

5.2.2.2 Adversary model

The adversary who wants to steal secrets of users is assumed to have the following capabilities.

A_1 The adversary can compromise many nodes (computers of users). Compromised nodes know and collaborate well with each other to achieve their goal: to steal as many secrets as possible.

A_2 Delegates of a user maybe public and thus may also be known to the adversary.

A_3 The adversary has computational power to perform dictionary-attacks to decrypt the shares if she obtains it and can reconstruct the secret successfully. Furthermore, the adversary is cost-insensitive and has as much time as she wants to complete the attacks.

A_4 A user losing her secret may be infected and under control of the adversary. This leads to the spreading of infection (more peers become a bad choice as delegates) that may contaminate the whole network.

The assumption A_2 is to enable easy reconstruction of the secret without requiring each secret owner to remember her list of delegates. On the negative side, it reduces the attack cost of an adversary: she may not need to probe many users to steal a specific key. A practical system can put a few extra safe-guards such as not making the delegates public knowledge, at the cost of increased system design complexity. In summary, the above adversary is extremely powerful and my security analysis will be done under such pessimistic assumptions.

5.2.2.3 Security of the threshold-based secret sharing

The security and correctness of a (k, n)-threshold-based secret sharing protocol depend on the number of *trustworthy* and *corrupted* delegates in the delegate selection. A *trustworthy delegate* sends correct shares only to authorized requesters. Additionally, trustworthy delegates do not steal the secret by colluding with others to steal the original secret. A delegate who is not trustworthy is defined as *corrupted* or *malicious*.

Given the above adversary model, it is possible for an adversary to decrypt any encrypted share. Thus a (k, n)-secret-sharing scheme is secured if and only if there are: (1) at most $k-1$ bad delegates; and (2) at least k trustworthy delegates available in a user's delegate selection to restore the secret. Therefore, the remaining and most important concern of a user is to choose her delegates to prevent corrupted delegates from stealing her secret by colluding with others. An approach to this problem is proposed and analyzed in Section 5.2.4.

A number of other assumptions are also made in the subsequent analysis:

- A message from the owner to an off-line delegate can be pending. When online again, the delegate pulls all these off-line messages and processes them accordingly. Such a message storage feature can be implemented by any decentralized storage mechanism. For example, in PeerSON (Buchegger *et al.*, 2009) the feature is implemented by using an implementation of a DHT lookup service (Rhea *et al.*, 2005).
- It is assumed that the compromise of a node by an adversary does not jeopardize its availability, similar to (Xu *et al.*, 2008). This is realistic since if infected machines become unavailable, e.g., cannot boot up or cannot connect to the network, this can signal to the owner to scan and clean her computer from malicious software.

5.2.3 Problem formulation and a preliminary analysis

The delegate selection of different users in the DOSN implicitly establish a following directed delegate graph $G = \langle \mathcal{U}, \mathcal{E} \rangle$, where:

- $\mathcal{U}$ is the set of users in the DOSN
- $\mathcal{E} = \{(u, v), u, v \in \mathcal{U}\}$ is the set of directed edges of the graph, in which the edge (u, v) implies that u selects v as one of his or her delegates.

Fig. 5.7(a) shows the illustration of a social graph. The delegate graph given in Fig. 5.7(b) is constructed on top of the social graph and implies that, for example, node B uses A and D as its delegates.

Define $N(i) = \{j : (i, j) \in \mathcal{E}\}$ as the set of delegates of i, and let $0 < \xi = n/k \leq 1$ be the threshold of the secret sharing scheme. Denote $I(t, G), t = 0, \ldots, T$ be the set of nodes in the delegate graph G that are under control of the adversary at time t, where $\mid I(0, G) \mid= c$ is the number of nodes under control of the adversary at the beginning, or its initial cost. For each node $i \in \mathcal{U}$, let the binary variable $x_i(t) = 1$ if $i \in I(t, G)$ and 0 otherwise. By controlling the nodes $I(t, G)$, the adversary has access to the shares kept by these nodes. At time $t + 1$, the

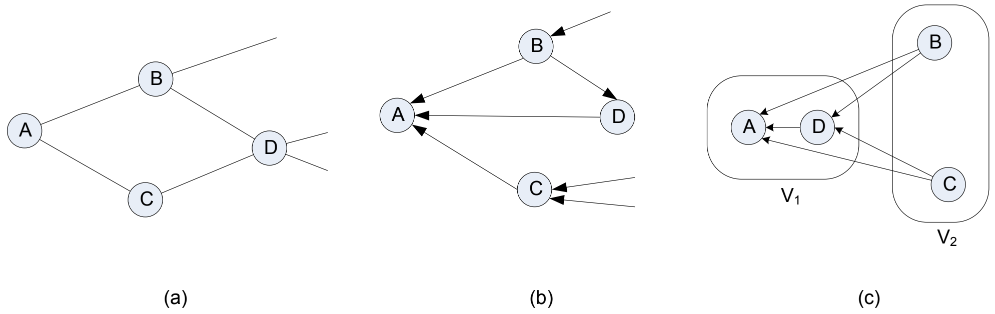


Figure 5.7: (a) Social graph; (b) Delegate graph; (c) Example in Problem 2.

adversary can steal all keys of those nodes $I(t+1,G)$ where:

$$I(t+1,G) = I(t,G) \cup \{i \in \mathcal{U} : (x_i(t) = 0) \wedge (\sum_{j \in N(i)} x_j(t) > \xi \mid N(i) \mid)\} \tag{5.2}$$

Therefore, the most powerful attack of the adversary is the solution of the following optimization problem

Problem 1 *Find $I(0,G)$ to maximize $\mid I(T,G) \mid$, subject to a fixed cost $\mid I(0,G) \mid= c$.*

Note that in the above notations, $T > 1$ means the case where infection may spread, i.e., a node losing the key eventually becomes under control of the adversary. $T = 1$ means there is no such spreading of infection.

Consider a special case of Problem 1, for $T = 1$, $\xi = 1$. We will show that solving this small problem is already difficult. A simple proof of this is given as follow.

First, we consider the following two related problems.

Problem 2 *Consider the directed delegate graph $G = \langle \mathcal{U}, \mathcal{E} \rangle$. For a given integer number $s <\mid \mathcal{U} \mid$, find $V_1, V_2 \subseteq \mathcal{U}$ where $\mid V_2 \mid= s, V_2 = inneighbors(V_1) = \{u : (u,v) \in \mathcal{E}, v \in V_1\}$, such that $\mid V_1 \mid$ is minimized.*

Problem 3 *The vertex expansion $c_v(G)$ of an undirected graph $G = \langle V, E \rangle$ is defined as:*

$$c_v(G) = \min_{X \subseteq V, 1 \leq |X| \leq |V|} \frac{\mid N(X) \mid}{\mid X \mid} \tag{5.3}$$

In Problem 2, V_1 is the minimal set of nodes the adversary must control initially so that it can steal s keys of a given set of nodes (V_2), as illustrated in Fig. 5.7(c).

Second, we will show that the special case of Problem 1 with $T = 1$, $\xi = 1$ can not be solved in polynomial time by contradiction. In fact, suppose that the Problem 1 with with $T = 1$, $\xi = 1$ can be solved in polynomial time, we can find $max \mid I(T,G) \mid$ for a given c, and $T = 1, \xi = 1$ in polynomial time. This solution can then be used to solve Problem 2 easily, also in polynomial time. In fact, we may find $max \mid I(T,G) \mid$ for each $c = 1$ to $\mid V_2 \mid$. The solution of Problem 2 is the set $V_1 = I(0,G)$ such that $max \mid I(T,G) \mid=\mid V_2 \mid= s$.

Next, it is observed that Problem 2 reduces to the computation of the vertex expansion of an undirected graph (Problem 3). Consider a special directed delegate graph where each delegate v of a node u also uses v as a delegate, i.e., $(u, v) \in \mathcal{E}$ implies that $(v, u) \in \mathcal{E}$. Thus this special delegate graph G is equivalent to an undirected graph. For this graph G, the solution of Problem 2 gives us two sets V_1, V_2, where $N(V_2) = V_1$. For each $s = 1$ to $\mid V \mid$, we can use the solution of Problem 2 to find the minimal set V_1 such that $\mid V_2 \mid = s$. By doing this we eventually compute the vertex expansion $c_v(G)$ trivially: $c_v(G)$ is simply the minimum of the ratio $\mid V_1 \mid / \mid V_2 \mid$ over $1 \leq s \leq \mid V \mid$.

Therefore, Problem 3 can be solved in polynomial time. This is in contrast with what is known about Problem 3: it is NP-hard and there is no known approximate algorithm to find a solution with up to a constant approximation factor (Alon, 1998; Bui & Jones, 1992). As a result, Problem 1, for general T and ξ can not be solved in polynomial time and there exists no good approximate algorithm for it.

Consequently, the best delegate selection algorithm is to build the optimal delegate graph $G = \langle \mathcal{U}, \mathcal{E} \rangle$ among all possibilities $\mathbb{G}$ such that:

$$G = argmin_{G \in \mathbb{G}} max \mid I(T, G) \mid \tag{5.4}$$

where $I(T, G)$ is defined for a given delegate graph G as in Equation (5.2).

Since the problem of finding the optimal attack strategy $max \mid I(T, G) \mid$ of a delegate graph G is hard, it is also very difficult to develop the theoretically-optimal delegate selection strategy according. As a result, we need to investigate different heuristics to select the delegates and to construct the adversary attack strategy accordingly. In the following section we study a possible greedy solution to select the most reliable delegates and perform various experiments to test its effectiveness against different attack strategy of the adversary.

Another approach is to study robustness of different topologies of a delegate graph under various adversarial attacks to identify the most attack-resilient topologies. Then, users in the social network would collaboratively construct an overlay delegate graph on top of the social graph. Available decentralized topology construction algorithms can be used for this purpose. This is another promising direction I have yet to study in this work.

5.2.4 Selection of reliable delegates: a heuristic approach

Due to various privacy and security settings, it is generally impossible for a user to crawl the whole network and gather all important information to best select the delegates. For example, personal data of a user and her relationships with others in most cases are not publicly available. Therefore, a user can only use her local knowledge and available public information in the network when selecting the delegates. Such a selection approach is formally described as follows.

Consider a DOSN $\langle \mathcal{U}, f, \mathcal{RP} \rangle$ as in Def. 13. Denote as $\mathcal{F}_u = \mathcal{F}_u^1$ the set of direct friends of a user u. The set of k-hop-away friends of u, where $k > 1$ is recursively defined as: $\mathcal{F}_u^k = \{w \mid w \in \mathcal{F}_v, v \in \mathcal{F}_u^{k-1}\}$. The set of all indirect friends of u is $\mathcal{F}_u^\infty = \bigcup_{k=1}^\infty \mathcal{F}_u^k$.

Let $\mathcal{P}_u^\infty$ be public personal information of users in $\mathcal{F}_u^\infty \subseteq \mathcal{U}$ and denote as f_u^∞ the set of connections among them. A (personalized) algorithm for a user u to select her delegates is given in Def. 14.

Definition 14 *A* delegate selection *of a user u is defined as an algorithm operating on the user's personalized view $\langle \mathcal{F}_u^\infty, f_u^\infty, \mathcal{P}_u^\infty \rangle$ on the social network and outputs a list of delegates $\mathcal{D}_u \in 2^{\mathcal{F}_u^\infty}$.*

Let $\mathcal{D}$ be a set of delegates selected by u for a (k, n)-secret-sharing scheme[1]. Denote as $\mathcal{D}_c$ and $\mathcal{D}_a$ be respectively the number of corrupted and available delegates in the set $\mathcal{D}$. Also, define $\mathcal{D}_{at}$ the number of trustworthy and available delegates in $\mathcal{D}$. I will present in this section a heuristic-based delegate selection approach via the use of a trust model that evaluates trustworthiness of each individual delegate candidate. This approach relies on the following concept of ε-security, given in Def. 15.

Definition 15 (ε-security) *The selection $\mathcal{D}$ is said to be ε-secured (to the secret owner) if and only if the probability that at least k trustworthy delegates are available in $\mathcal{D}$ is $Pr(\mathcal{D}_{at} \geq k) \geq 1 - \varepsilon$, where $0 < \varepsilon < 1$.*

We want to study the way u selects her set of delegates $\mathcal{D}$ to ensure the availability and secured access to her backed-up secret. For simplicity, fix the number k and the security parameter $0 \leq \varepsilon \leq 1$. Let $\mathcal{D} = \{i, 1 \leq i \leq n\}$. We want to select $\mathcal{D}$ based on k, ε such that the resulting (k, n)- secret sharing scheme is ε-secured (Def. 15).

Denote $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ and $A(n, n-2k+1) = Pr(\mathcal{D}_a \geq 2k-1)$= Pr (at most $n - 2k + 1$ are offline). Assume that the probabilities that a user i is trustworthy and available, are $0 \leq t_i \leq 1$ and $0 \leq a_i \leq 1$, respectively. One can verify that $T(i, 0) = \prod_{j=1}^{i} t_j$ and $T(i, i) = 1$. The following recurrence relations can be obtained using basic probability update rules:

$$T(i+1, l+1) = t_{i+1} T(i, l+1) + (1 - t_{i+1}) T(i, l), \;\; 1 \leq i \leq n \tag{5.5}$$

Similarly, for $1 \leq i \leq n$, $A(i, i) = 1$, $A(i, 0) = \prod_{j=1}^{i} a_j$, and:

$$A(i+1, l+1) = a_{i+1} A(i, l+1) + (1 - a_{i+1}) A(i, l) \tag{5.6}$$

Given our adversary model in Section 5.2.2.2, availability and trustworthiness of nodes are assumed to be independent. Therefore, the probability of at least k trustworthy delegates available among n delegates is:

$$\begin{aligned} Pr(\mathcal{D}_{at} \geq k) &\geq Pr(\mathcal{D}_a \geq 2k-1, \mathcal{D}_c \leq k-1) \\ &= A(n, n-2k+1) T(n, k-1) \end{aligned} \tag{5.7}$$

Proposition 4 gives us some properties of the probabilities $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ and $A(n, n-2k+1) = Pr(\mathcal{D}_a \geq 2k-1)$.

Proposition 4 *For any $n \geq k > 0$:*

[1] the index u is omitted for presentation clarity

(i) Irrespective of the selection $\mathcal{D}$, $T(n, k-1)$ and $A(n, n-2k+1)$, where $n > 2k-1$, are increasing functions of k and decreasing functions of n.

(ii) $T(n, k-1)$ is maximized where $\mathcal{D}$ is n delegates with highest trustworthiness t_i among the candidates.

Proof 9 *(i) We prove the monotonicity of $T(n, k-1)$ first. The proof for $A(n, n-2k+1)$ can be done similarly. From its definition, it follows that $T(n, k) \geq T(n, k-1)$. The proof of $T(n+1, k) \leq T(n, k)$ is also trivial. In fact, $T(n+1, k) - T(n, k) = t_{n+1}T(n, k) + (1 - t_{n+1})T(n, k-1) - T(n, k) = (1 - t_{n+1})(T(n, k-1) - T(n, k)) \leq 0$, since $T(n, k-1) \leq T(n, k)$ from the above result.*

(ii) Suppose that the delegate set $\mathcal{D}$ has n delegates, where each $i \in \mathcal{D}$ has $t_i \geq t_j, j \in [N] \setminus \mathcal{D}$.

Consider another delegate set $\mathcal{D}'$ different from $\mathcal{D}$ by one delegate. Define $\mathcal{D}' = succ(\mathcal{D}) = (\mathcal{D} \setminus \{i\}) \cup \{j\}$, where $j \in [N] \setminus \mathcal{D}$ and $t_i \geq t_j, 1 \leq i \leq n-1$. We will show $Pr(\mathcal{D}_c \leq k-1) \geq Pr(\mathcal{D}'_c \leq k-1)$.

As delegates have equal roles, the order of putting a user i in a delegate set D does not influence the probability $Pr(\mathcal{D}_c \leq k-1)$. Let us select the user i at the last step to form the set $\mathcal{D}$ and select j at the last step to form $\mathcal{D}'$.

It follows that $Pr(\mathcal{D}_c \leq k-1) = t_iT(n-1, k-1) + (1-t_i)T(n-1, k-2)$ and $Pr(\mathcal{D}'_c \leq k-1) = t_jT(n-1, k-1) + (1-t_j)T(n-1, k-1)$. Since $T(n-1, k-1) \geq T(n-1, k-2)$ according to (i), and $t_i \geq t_j$ from the definition of $\mathcal{D}'$, we have:

$$\begin{aligned} Pr(\mathcal{D}_c \leq k-1) - Pr(\mathcal{D}'_c \leq k-1) &= (t_i - t_j)(T(n-1, k-1) \\ &\quad -T(n-1, k-2)) \\ &\geq 0 \end{aligned}$$

More generally, given any selection $\mathcal{D}''$ different from $\mathcal{D}$ by $J \leq n$ users, one can verify that there exist a series of delegate selection $\mathcal{D}^j, 1 \leq j \leq J$ such that $\mathcal{D}'' = \mathcal{D}^J, \mathcal{D}^{j+1} = succ(\mathcal{D}^j), 1 \leq j \leq J-1$, and $\mathcal{D}^1 = succ(\mathcal{D}) = \mathcal{D}'$. Since $\mathcal{D}' = succ(\mathcal{D})$ implies $Pr(\mathcal{D}'_c \leq k-1) < Pr(\mathcal{D}_c \leq k-1)$, it follows that $Pr(\mathcal{D}''_c \leq k-1) < Pr(\mathcal{D}_c \leq k-1)$. Therefore, $Pr(\mathcal{D}_c \leq k-1) = T(n, k-1)$ is maximized with the delegate set $\mathcal{D}$.

5.2.4.1 Measuring trustworthiness of delegate candidates

Let i be a possible delegate candidate for u. In practice, i may be a friend of u, or a third-party provider offering data storage services. The measurement of the trustworthiness t_i of a user i is non-trivial. In our scenario, the notion of trust between two users is beyond the social trust between people, since the computer of a highly reliable and trustworthy friend may still be compromised by an adversary without the friend's awareness. Therefore, a user needs a more appropriate measure to evaluate the trustworthiness of a user before selecting her as a delegate. More precisely, t_i *is the personal belief of u on whether i is likely to be controlled by an adversary.* Such a value t_i depends on the following influential factors:

- *whether the node i is a well-known trusted entity*. For example, nodes from third-party providers offering data storage services can be seen as less vulnerable as they are usually equipped with up-to-date security patches and latest virus definitions.
- *whether i has potential to collude and steal a secret*, since curious friends may collude to get illegitimate access to unauthorized data. Also, if a user's friend is compromised by an adversary, other friends of hers are also vulnerable to attacks by the same adversary. In practice, viruses are likely to spread from one friend to another since people generally trust files or links sent by their friends. To minimize the influence of such attacks, we should give less trust to those delegates i with more chances to collude with each other. Informally, we should select delegates from different sets of friends to reduce the possibility of a collusive attack and minimize the influence of such a collusion.
- *whether i is an attractive target for an adversary*: since an adversary can minimize her attack cost by compromising nodes holding more keys, i is more attractive to an adversary if she is a delegate of many users. Hence, less trust should be put on those candidates i currently keeping more shares.

Given the above observations, the following heuristic is suggested to evaluate the trustworthiness of a user. The key idea is to explore social relationships among users to prevent the spreading of infection from malicious nodes already under adversary control, as well as reducing the chance of colluding among these malicious nodes. More concretely, we define:

$$t_i = \begin{cases} 1 & \text{if } i \in D^0 \\ 1 - \delta(b_i) - \sigma(l_i) & \text{if } i \in [N] = \mathcal{F}_u \setminus D^0 \end{cases} \tag{5.8}$$

where $b_i =\mid \mathcal{F}_i^{\infty} \cap \mathcal{F}_u \mid$ and l_i is the number of shares held by i. The candidate delegates $[N] = \{i, 1 \leq i \leq N\}$ are from the set of friends of the user $\mathcal{F}_u$. In Equation (5.8), D^0 is a set of $m < k$ preferred/trusted delegates chosen by the secret owner u. For instance, a good choice of D^0 include nodes from third-party providers offering data storage services.

Note that $b_i =\mid \mathcal{F}_i^{\infty} \cap \mathcal{F}_u \mid$ is the number of all indirect friends of i who are also in $\mathcal{F}_u \setminus D^0$. Thus $0 \leq \delta(b_i) \leq 1$ quantifies two influential factors: (1) how i is able to collude with her friends to compromise a shared secret; and (2) the influence i and her friends may have on the owner u if i is compromised by an adversary. Thus u puts higher trust on a friend i with less connection with others of her friends. Given this observation, in the delegate selection algorithm, I set $k > \max_{i \in [N]}\{b_i\}$ (Algorithm 7).

The term $0 \leq \sigma(l_i) \leq 1$ determines the attractiveness of i to an adversary, e.g., how many keys the adversary gets by compromising i. Note that $0 \leq \delta(b_i) \leq 1$ reach the maximal value of 1 at $b_i \geq b^* = k - m$ and be at the minimum 0 at $b_i = 0$. Similarly, $\sigma(l_i)$ is defined to achieve the maximal value of $1 - \delta(b_i)$ at $l_i = l^* = \max\{l_i, i \in [N]\}$ and be 0 at $l_i = 0$. That means currently we care more on the impact of $\delta(b_i)$ than of $\sigma(l_i)$. The testing of different impact weights by $\delta(b_i)$ and $\sigma(l_i)$ is subject to future work. Depending on user's prior belief on the environment vulnerability, the following functions can be used:

- *exponential*: e.g., $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{2^{b_i}-1}{2^{b^*}-1}$ and $\sigma(l_i) = (1 - \delta(b_i)) \frac{2^{l_i}-1}{2^{l^*}-1}$. These functions[1] are appropriate for *vulnerable* environments with potentially a lot of malicious users. Hence, the possibility that a node j would be compromised increases exponentially with the number of shares she keeps l_i and the number of common friends b_i between i and u.

- *logarithmic*: e.g., $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{log(b_i+1)}{log(m+1)}$ and $\sigma(l_i) = (1 - \delta(b_i)) \frac{log(l_i+1)}{log(l^*+1)}$. These functions are appropriate in *more secured* environments with fewer malicious users, thus the possibility j being compromised increases less than linearly with l_i and b_i.
- *linear*: $\delta(b_i) = 1_{\{b_i > b^*\}} + 1_{\{b_i \leq b^*\}} \frac{b_i}{b^*}$ and $\sigma(l_i) = (1 - \delta(b_i)) \frac{l_i}{l^*}$, which are applicable in *neutral* environments with a moderate number of malicious users.

5.2.4.2 A trust-based delegate selection algorithm

According to Proposition 4, the probability $T(n, k-1) = Pr(\mathcal{D}_c \leq k-1)$ is maximized by choosing $\mathcal{D}$ as n delegates with highest trustworthiness t_i from the candidates $[N]$. It is assumed that messages to unavailable delegates can be kept pending and processed later on when they go online (Section 5.2.2). Hence the unavailability of delegates do not affect the computation of the probability $T(n, k-1)A(n, n-2k+1)$, and we can approximate $T(n, k-1)A(n, n-2k+1) = T(n, k-1)$.

With the trust measure in Section 5.2.4.1, the selection of delegates to maximize the probability $T(n, k-1)$ is given in Algorithm 7. Roughly speaking, we sequentially pick a user i with the highest trust value t_i from remaining candidate delegates. Thus Algorithm 7 forms a delegate set $\mathcal{D}$ approximately maximizing $T(n, k-1)$. Since $Pr(\mathcal{D}_{at} \geq k) \geq T(n, k-1)$, we expect this algorithm to maximize the probability that the delegate set achieved the desired security level ε.

Algorithm 7 stops in two cases: first, all available N candidates are selected to be delegates; second, we achieve the desired security level $1 - \varepsilon$. In the first case, the actual security level of the selection is $Pr(\mathcal{D}_{at} \geq k) \geq Pr(\mathcal{D}_c \leq k-1) = T(n, k-1)$. This lower bound $T(n, k-1)$ also reflects the vulnerability of the environment and may be used by the user to decide whether to share her secret. In later experiments, a user backs up her secrets only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$ is a parameter of the experiments.

5.2.5 Experiments

The DOSN is simulated as a discrete event-based system with the agent-based simulation toolkit Repast (North *et al.*, 2006). A social network in an experiment is a graph of of 1028 users. The network topology follows Watts-Strogatz small world model (Watts & Strogatz, 1998), with connection radius 2 and rewiring probability 0.8 (thus this network has short average path length and a small clustering coefficient). This model was due to its simplicity and small world properties of social networks. Larger scale simulations are possible, yet smaller networks

[1] The function $1_{\{A\}}$ evaluates to 1 if A is true and to 0 otherwise.

Algorithm 7 *selectDelegates*(candidates $[N]$, trustworthiness t_i for each $i \in [N]$, threshold k, security parameter ε): selected delegates $\mathcal{D}$, achieved security level $T[n, k-1]$

```
$n = 0$; $\mathcal{D} = \varnothing$; $\mathcal{L} = [N]$; /* $\mathcal{L}$ is the current candidate list */
$T[n, 0] = T[n, 1] = 1$;
while ($n \leq N$ and $T[n, k-1] < 1 - \varepsilon$) do
    Pick user $i$ with highest $t_i$ in $\mathcal{L}$;
    $n = n + 1$; $\mathcal{D} = \mathcal{D} \cup \{i\}$; $\mathcal{L} = \mathcal{L} \backslash \{i\}$;
    $T[n, 0] = t_i T[n-1, 0]$; $T[n, k] = 1$;
    for $l = 1$ to $\min(n-1, k-1)$ do
        $T[n, l] = t_i T[n-1, l] + (1 - t_i) T[n-1, l-1]$;
    end for
end while
Return $\mathcal{D}$ and expected lower bound of security level $T[n, k-1]$;
```

were used to reduce the simulation time. Another reason is that a DOSN is likely to be self-organized in ad-hoc manner, e.g., among people within a geographical vicinity. Hence its size should be much less than traditionally centralized social networks. Experiments with other types of topologies are subject to future work (this is in fact a limitation of this work). A message-passing system was implemented where messages arriving at an unavailable recipient are made pending and processed later when the recipient goes back online. Such properties can be realized in practice with distributed storage techniques: messages are encrypted and stored in a DHT look up system (Stoica *et al.*, 2001) built on top of the user's computers. User unavailability hence, does not affect their being selected as delegates: a user sending a request for a share to an off-line delegate simply waits till the latter is available.

5.2.5.1 Implementation of delegate selection approaches

Different delegate-selection approaches were implemented, each of which uses only local knowledge to select delegates, as in Def. 14.

- FRIENDBASED: a user selects all her friends as delegates. This approach was studied in a previous related work (Xu *et al.*, 2008) for a different application (document signing) and in a limited case (without spreading of infection from malicious users).
- RANDOMWALK: a user random walks on the (public) social network graph and picks a delegate after $TTL = 7$ steps. This approach selects those delegates connected with many friends who also have high connectivity, e.g., nodes with higher PageRank-like values.
- TRUSTBASED: the trust-based delegation selection algorithm described in Algorithm 7, Section 5.2.4. I used a reasonably small security parameter $\varepsilon = 0.001$, and with $\sigma(.)$ and $\delta(.)$ as exponential functions, i.e, users believe the environment is vulnerable. A user only decides to share a secret if and only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$ is an experiment parameter.

5.2.5.2 Experiment design

The system was tested with two following performance metrics: f_L and f_b. $0 \leq f_L \leq 1$ is the fraction of secrets (private keys) lost among those backed up by using the threshold-based secret sharing scheme. A smaller f_L implies a more secured system and thus is preferable. $0 \leq f_b \leq 1$ is the fraction of users (among all) who can back up their secrets successfully. A higher f_b means higher usability of the system and thus is better. Note that f_b is only meaningful to the TRUSTBASED selection algorithm, since only this algorithm requires that a secret is backed up only if the achieved security level is $Pr(\mathcal{D}_{at} \geq k) \geq \tau$, where $0 \leq \tau \leq 1 - \varepsilon$. Other selection approaches (FRIENBASED or RANDOMWALK) have $f_b = 1$. The TRUSTBASED approach was tested with different values of τ, for which in most cases $f_b > 0.9$, and τ does not clearly affect f_L. Therefore, I will focus on the performance metric f_L in later experiments with a given value $\tau = 0.75$.

Given the above metrics, the system load model consists of the following factors:

- *inf*: whether there is an infection spreading in the network, i.e., whether a user having lost her secret also becomes under adversary control.
- *pmal*: the percentage of initially malicious users, i.e., the number of users already under adversary control at the beginning of the simulation before the adversary commands them to initiate the coordinated attack. A higher *pmal* means a more vulnerable environment.
- *adv*: the distribution of the initially malicious users in the network. The distribution can be random or based on certain criteria, e.g., number of friends of a user.

Beside the load factors and their intensities, the major factors influencing the system performance are:

- *select*: the algorithm to select delegates for the secret sharing protocol, which includes the three algorithms FRIENDBASED, RANDOMWALK, and TRUSTBASED as described in Section 5.2.5.1.
- $0 \leq \xi \leq 1$: the threshold k/n of a (k,n)-threshold secret-sharing scheme being used.

The goal of the experiments is to measure the effects of two factors *select* and ξ to the system performance f_L under various load intensities *inf*, *pmal*, and *adv*.

Each simulation was run with appropriate parameters and each performance metric was measured when the simulation reached the stationary regime. The result (not shown) confirms that the distribution of f_L is approximately Gaussian and perfectly iid. Therefore, for later experiments, I only ran each simulation with N=35 replications (sufficiently large sample size) and summarized the measurements of f_L with its means and confidence intervals at level 95%. As data is roughly normal iid, this summarization shows both accuracy and variability of the obtained results.

5.2.5.3 Effects of the threshold values ξ

The influence of the cryptographical threshold $\xi = k/n$ to system security was measured under various load intensities. For this goal ξ was varied from 0.1 to 1.0 for each of the delegate selection approaches FRIENDBASED, RANDOMWALK, and TRUSTBASED. The influence of ξ on the fraction of secrets lost f_L for each selection algorithm is given in Figure 5.8, where the Y-axis shows the mean of f_L and 95%-confidence intervals of the mean of f_L.

For each given delegation-selection algorithm, the load intensity was varied as follows. First, *pmal* was increased from 0.1% to 50% to simulate environments with different numbers of malicious users under adversary control. To reduce the simulation parameter space, the distribution of these initially malicious users in the network was randomized (*adv*=RANDOM). Each experiment was carried out with both cases (*inf* = *false* and *inf* = *true*).

Generally f_L is expected to be lower with higher values of ξ. In fact, two main observations can be drawn from the results in Figure 5.8. Firstly, for any delegate-selection algorithm and for any case of infection spreading, the smallest fraction of keys lost f_L is achieved with $\xi = 1$. This is intuitive: given a secret sharing scheme with $\xi = 1$, the adversary must successfully attack all delegates of a user to be able to steal the victim's secret. For values of $\xi < 1.0$ the differences are not substantial.

Secondly, the system security with an infection spreading (left pane of each figure) is much lower than without infection spreading (right pane). Figures 5.8(a, c, e), left panes, show that the adversary can steal a large fraction of secrets (from 50% up to 100% of secrets in most cases) by initially controlling a small number of malicious users (from 0.2% if $\xi < 1$ and from 5% if $\xi = 1$). The influence of the infection spreading is minimal in two cases: (1) with $\xi = 1.0$ and the delegate-selection approach FRIENDBASED as in Figure 5.8(a, left), and (2): for the TRUSTBASED delegation selection algorithm, as in Figure 5.8(left side of e, f). These observations will be verified in the next section.

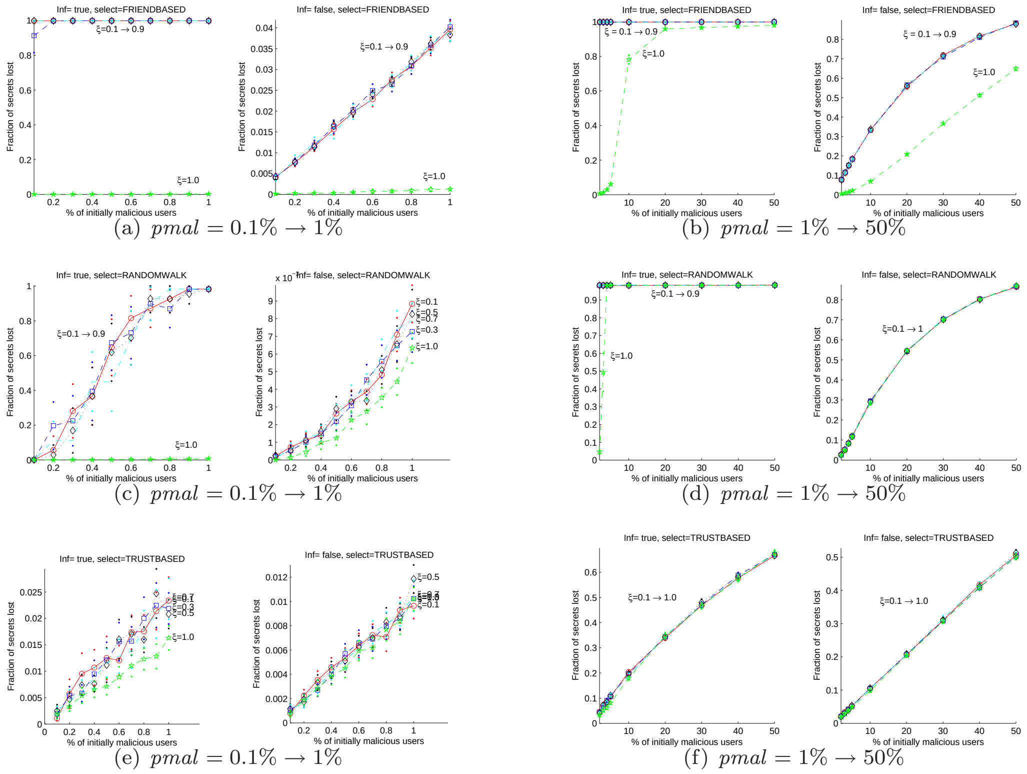
5.2.5.4 Effects of delegate selection algorithms

As the next step, effects of different delegate-selection algorithms (FRIENDBASED, RANDOMWALK, and TRUSTBASED) to the system security were measured under various load intensities.

Specifically, the first load factor *pmal* was varied from 0.1% to 50% to simulate different environments with small (0.1% to 1%) and large (1% to 50%) numbers of malicious users under adversary control. Initially malicious users were randomly distributed in the network and each experiment was carried out in both settings: with or without infection spreading (*inf* = *false* or *true*).

From previous experiments (Section 5.2.5.3), only two critical values ($\xi = 1$ and $\xi < 1$) of threshold ξ are of interests. The results are given in Figures 5.9.

It is observed that the selection approach TRUSTBASED is the best or comparable with the best in most cases, as shown in Figure 5.9(a, b, d). In the only case with a very small number of initially malicious users, and with $\xi = 1$ of Fig. 5.9(c), the selection algorithm FRIENDBASED is the best. In practice, using the maximal threshold $\xi = 1$ may not be preferable, as the secret owner may have to wait for every delegate to be online to reconstruct her backed-up secret. One possible reason why the TRUSTBASED selection algorithm performs less well in this case is the difference between the actual vulnerability of the environment and the use of exponential functions $\sigma(.)$ and $\delta(.)$ to evaluate the trustworthiness measure of available delegate



(a) $pmal = 0.1\% \rightarrow 1\%$

(b) $pmal = 1\% \rightarrow 50\%$

(c) $pmal = 0.1\% \rightarrow 1\%$

(d) $pmal = 1\% \rightarrow 50\%$

(e) $pmal = 0.1\% \rightarrow 1\%$

(f) $pmal = 1\% \rightarrow 50\%$

Figure 5.8: Influence of the crypto threshold ξ on the fraction of secrets lost f_L with different delegate-selection algorithms. The two critical values of ξ are $\xi = 1.0$ and $\xi < 1.0$

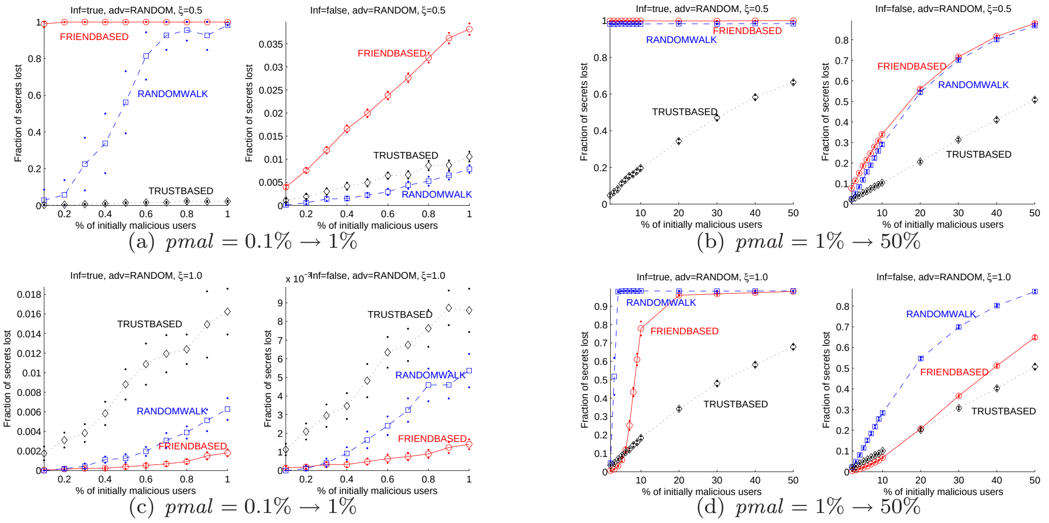
candidates. Nevertheless, performance of the TRUSTBASED algorithm is still reasonably good in this case: $f_L < 0.018$ for $pmal \approx 1\%$ and $inf = true$ as in Fig. 5.9(c). Remember that f_L is the system performance metric measured under a very pessimistic estimation, where the adversary had unlimited computational power to decrypt any protected share it ever stole, the attained performance is acceptable.

The RANDOMWALK delegate selection approach, though intuitively appealing, is not superior. Its performance is somehow better where there is no infection spreading and the adversary controls a very small number nodes, as shown in Figure 5.9(c) and in the right pane of Figure 5.9(a).

Considering different adversary distributions

Another interesting question is how well different delegate selection algorithms perform under different adversary distributions. Specifically, we may want to know whether the adversary may exploit her knowledge of the current delegate selection of users to focus her attack in a number of well-chosen users in the system.

Since the delegate selection algorithms FRIENDBASED, RANDOMWALK, and TRUST-



(a) $pmal = 0.1\% \rightarrow 1\%$ (b) $pmal = 1\% \rightarrow 50\%$

(c) $pmal = 0.1\% \rightarrow 1\%$ (d) $pmal = 1\% \rightarrow 50\%$

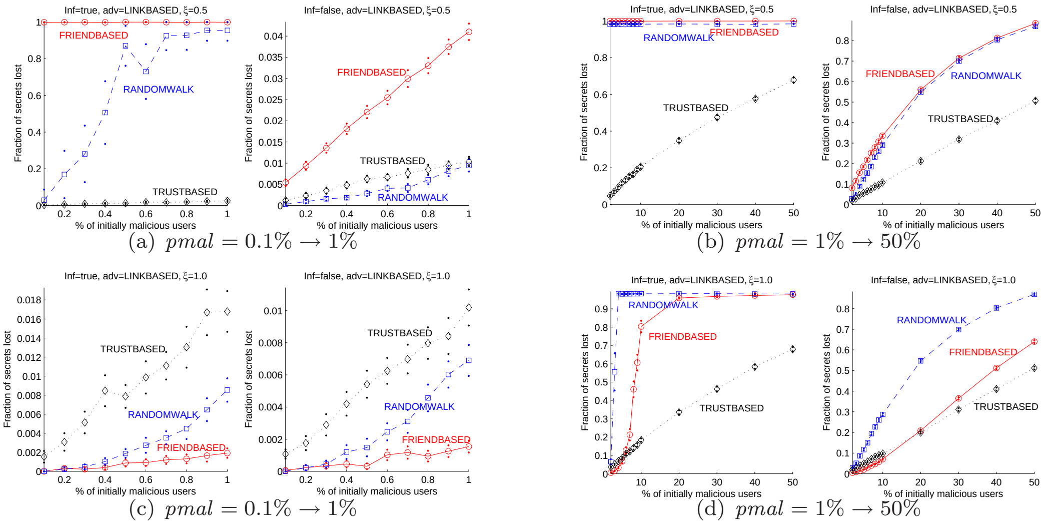
Figure 5.9: Influence of delegate selection algorithms on f_L, distribution of $pmal$=RANDOM, $\xi = 0.5$ (a,b) and $\xi = 1$ (c,d)

BASED all use connectivity of users as a selection criteria for a delegate, an adversary may attempt the following attacks:

- LINKBASED: initially, the adversary tries to control nodes with higher numbers of links. These nodes are likely to be chosen by many users as their delegates, thus controlling these nodes help the adversary to steal more secrets at a lower cost.
- KEYBASED: the adversary focuses on compromising nodes currently keeping higher numbers of shares. This attack is even more powerful, as apparently controlling these nodes gives the adversary the highest likelihood of stealing the maximal number of secrets.

Various experiments were also performed to measure the influence of the above adversarial attack strategies. Load intensities are set up as in previous experiments, with varied numbers of initially malicious users: $pmal$ varied from small (0.1% to 1%) to larger (1% to 50%). Both cases of ($inf = false$ or $true$) are considered, each with two representative threshold values: $\xi = 0.5$ and $\xi = 1$.

Figure 5.10 shows the performance of different delegate-selection approaches under the adversary attack using the LINKBASED distribution. The result is similar to the previous case. The selection approach TRUSTBASED still performs either best or comparable to the best approach in almost all cases. Similar to the previous experiment with randomly distributed adversary, the TRUSTBASED algorithm performs less well than the FRIENDBASED approach where there is a very small number of initially malicious users and the threshold is $\xi = 1$. The algorithm RANDOMWALK is not superior in majority of cases we studied. Experiments for the adversary distribution KEYBASED are more complicated, as they requires detailed time-variant modeling of distributions of adversarial attacks and the selection of delegates. This experiment is subject to future work.



(a) $pmal = 0.1\% \rightarrow 1\%$

(b) $pmal = 1\% \rightarrow 50\%$

(c) $pmal = 0.1\% \rightarrow 1\%$

(d) $pmal = 1\% \rightarrow 50\%$

Figure 5.10: Influence of delegate-selection algorithms on f_L, distribution of $pmal$=LINKBASED, $\xi = 0.5$ (a,b) and $\xi = 1$ (c,d)

5.2.6 Related work

Applications and usability of threshold cryptography in P2P and mobile ad-hoc networks are discussed in detail in (Saxena *et al.*, 2007). This work mostly focuses on evaluating computational and communications overhead of different threshold cryptographical protocols. Security properties of such protocols under collusive attacks of many malicious nodes are not discussed.

There are several improvements of the original threshold-based secret sharing scheme, e.g., produce verifiable shares, change the threshold dynamically, improve the protocol's computational and communication efficiencies (Schneier, 1995). The selection of reliable delegates to minimize the influence of collusive attacks, however, has not been the focus of the security research community. Various security solutions mostly protect secret shares by encrypting these shares with passwords. These works are complementary to my work, since we can use any enhanced threshold cryptographical approach for a delegated key recovery scenario.

To the best of my knowledge, my approach is the first one applying threshold cryptography to enable private key backup and recovery in distributed social networks. Another novelty of this work is the proposal of appropriate trust measures based on social relationships among users. This solution is shown to be effective against possible infection spreading, and resilient under presence of various malicious nodes with different distributions.

The work most related to this one is (Fran *et al.*, 2008; Xu *et al.*, 2008), where the authors apply threshold cryptographical approaches for a different problem on social networks. (Xu *et al.*, 2008) considers the application of threshold cryptographical protocols for signing documents, which is a different problem since it requires the secret owner to be available to issue the digital signature. Regarding the technical approach, my work is also different from (Fran *et al.*, 2008; Xu *et al.*, 2008). First, I study the effectiveness of many delegate selection approaches given the presence of various numbers of malicious users and with possible infection spreading. The FRIENDBASED delegate selection algorithm that we analyzed is the same as the approach of using trusted friends to store the cryptographical keys proposed by (Xu *et al.*, 2008). Second, the adversary model I am considering is more powerful with its capability of spreading its infection to many nodes. My simulations also consider many possible attack models of the adversary. (Fran *et al.*, 2008) proposes to create certificates based on agreement of a number t of nodes in the network determined dynamically. The main result is a preliminary analysis on the probability of nodes being attacked. The issues of infection spreading and the selection of reliable delegates, however, are not yet studied.

In a wider context, this work is also related to credential-based access control in P2P and social networks. Credentials can be generated by specialized hardware such as the Trusted Computing Platform (Sandhu & Zhang, 2005), or based on certain attributes of the requesters (Palomar *et al.*, 2008). Most work in this direction addresses the general problem of access control with the assumption that access is granted to authorized users with identities known beforehand. This assumption is not valid in the key backup and recovery scenario and thus these approaches are not applicable. In other works, such as (Carminati & Ferrari, 2008; Chakraborty & Ray, 2006; Domingo-Ferrer *et al.*, 2008; Palomar *et al.*, 2008), heuristic measures are used to evaluate trust between the resource owner and the requester before the credential is granted. For example, (Carminati & Ferrari, 2008) assigns each edge between users a trust level and a relationship type and uses a central node for registration and management. (Domingo-Ferrer *et al.*, 2008) proposes to grant access to resources based on chain of trust relationships between the resource owners and the requesters. An open problem of these solution approaches is a detailed analysis of their correctness and security under collusive attacks and spreading of adversarial infection in the network.

5.2.7 Conclusion and future work

The question of secure backup and recovery of secrets in (distributed online social) systems is an important and challenging issue. This work provides a promising first step toward an appropriate technical solution to this problem. I have studied the application of threshold-based secret sharing protocols for this purpose and suggested potential mechanisms to improve the security of these protocols. Specifically, I have proposed an algorithm to select the most reliable delegates to enable such a secure secret sharing in a network of untrusted nodes. This approach selects delegates based on a simple trust measure that exploits social relationships among the secret owner, delegate candidates, and their friends, to minimize the probability that a set of delegates collude to steal the secret. Such an approach has been shown to perform very well under a variety of scenarios, even with a large number of initially malicious users with possible spreading of infection in the network.

A limitation of this work is that it only considers a small-world network topology with homogeneous degree distribution. It would be important to also measure and analyze performance of various selection algorithms in larger scale heterogeneous networks, e.g., with heavy-tailed node degree distributions. The dynamics of networks with a growing number of users is also an important factor yet to be studied.

As part of future work, I plan to study realistic cost models of an adversary to integrate with the design of potential delegate selection mechanisms. This issue is important since the attack by a rational adversary may be well-related to its endured cost. The self-healing capability of the distributed social network via the detection and cleaning of malicious and infected nodes is also an interesting issue we plan to study as part of future work.

A potential approach to this problem is to rely on the existence of a number of highly immune nodes in the system. These nodes may actively scan their friends periodically to detect if these users are compromised by an adversary, e.g., by checking for spyware and malware on these nodes. Thus a compromised computer of a user can be detected and cleaned to become a trustworthy delegate. In practice, such users with high immunity can be Antivirus centers or (storage) service providers (which provide periodical virus definition updates and latest security patches), and whose connections can be their subscribed customers. I want to analyze the minimally optimal number of such immune nodes and their placement in the network such that it is most effective to defense users from various collusion attacks from an adversary.

Chapter 6

Open Reputation-based Trust Management Systems

Reputation-based trust management systems are omnipresent and increasingly become essential to several Internet applications, from e-trading systems, e.g., eBay.com or uBid.com to various online fora such as Epinions.com or Digg.com. However, architectures and implementations of these systems are typically closed: specifically, each system is developed to use its own approach to derive trust from reputation ratings given the available observation data collected from within its own user communities. Such a closure is not in accordance with the open nature of the Internet and its ecosystem. In practice, several similar applications have overlapping user bases, such as different trading sites on the same geographical region, and users move freely from one site to another. Furthermore, many trust management systems on related domains have similar working principles of deriving trust from analyzing user behaviors and activities, especially systems in the same or similar application domain with similar business processing logics. Therefore, the trust management systems being used by each system may use data from very related operational contexts and thus these systems may gain significant benefits from each other by sharing this data.

This chapter presents my on-going work related to the building of an open architecture model for reputation-based trust management systems. This open approach can be implemented and deployed by different applications across domains. I propose that similar systems can exchange and use their knowledge and information to benefit each other in many aspects. It is shown that under certain assumptions, the combination of different reputation-based trust management systems together may help each system achieving better performance in detecting misbehavior, thereby indirectly promoting more cooperation of users. I explore several possibilities and challenges related to this research direction, and provide an outline or an initial solution to these research questions.

Part of the work in this chapter is published in (Vu *et al.*, 2009b).

6.1 Introduction

As reputation is a well established means to determine trustworthiness in online systems, a large number of reputation-based trust systems have been proposed (Golbeck, 2006; Jøsang *et al.*, 2007; Wang & Lin, 2008). Several commercial initiatives have also been proposed based on the concept of assessing participants's trustworthiness and performance based on based on their reputation and historical actions, such as auction sites (eBay.com, ePier.com), recommender systems (Amazon.com), or e-mail spam fighting (TrustedSource, Trend Micro Email [1]).

However, typically these systems are *"closed"* to each others: very little information about the user communities and the reputation mechanisms being used are exchanged with other systems. For example, the set of participants in most cases is known only within one system. In addition, possible actions of users, the evaluation of their behaviors, and the mechanism to derive trust from such user interactions are predetermined in the system design. Hence, each system exploits the history of users' actions in a closed environment. This is in stark contrast to the open nature of the Internet where no firm system boundaries and stable specifications of actions and information exist. Existing information is hardly reused among systems and thus the trust evaluation on a newly joined identity is usually very difficult (the "cold start" issue).

Nowadays, new online communities dynamically emerge in various contexts through the exchange of views and services among systems. The trustworthiness of the members of these communities is evaluated from scratch. This is not *socially optimal*, as the members of these communities are often also members of existing communities, possibly employing different virtual identities, and their trustworthiness in these contexts has already been evaluated.

A reputation system in a closed community is able to recognize specific untrustworthy behavior patterns within that context, given the structure of the community, the structure of the reputation system, and the history of actions in the context. However, people actively participate in various online communities, such as social networks (e.g. LinkedIn.com, myExperiment.org, Facebook.com), online transaction environments or product review sites. *Reputation and behavior of a user in a system may well be related to his reputation and behavior in another.* Combining reputation information for the same members from different contexts may be more effective a mechanism for identifying *additional patterns* of malicious behavior within each individual context. Thus, reputation systems should take advantage of not only diverse information sources in the Internet, such as social networks, recommender systems, content sharing systems, or semantic search engines, but also other reputation systems.

This chapter sketches the design of the next generation, open reputation-based trust systems. The main design principle is to enable a system to exchange existing trust-related information such as reputation from different contexts across applications and system boundaries. The goal of this information sharing is to effectively enhance the learning of trustworthiness of participants. I provide a detail discussion on related challenges, namely identification, mapping of reputation semantics, contextual distances between application settings, reputation disclosure (dis)incentives and privacy. For example, the critical issue of identification can more effectively

[1] www.trendsecure.com/portal/en-US/tools/security_tools/emailid

be dealt with based on entity-matching and the social structure of different systems. It is argued and theoretically proven that even naïve combinations of reputation values from two different communities can result in a system capable of detecting misbehavior more effectively than the individual trust mechanisms themselves under certain conditions on trust and reputation semantics. This work is the first attempt to develop an open trust management system architecture and the first one systematically defines and studies the challenges and benefits of such an architecture for information sharing among different reputation-based trust systems.

6.2 Motivation and Opportunities

An open trust management system offers several advantages compared to existing systems with a closed architecture. Such advantages come from the fact that an open system may reuse much information from another. Generally there is much knowledge that a system may offer and benefits from another system, especially among those used for similar applications. For example, two commercial auction sites operating in a same geographical region such as *ricardo.ch* and *ebay.ch* may have a considerate overlapping user base. Sharing and exchanging identities of these common users may may help to discovery common user behavioral patterns in both system more effectively. It is expected that the reuse and combination of reputation and trust-related information across systems would give us benefits even beyond of them individually achieved by each system for several reasons:

1. The trust learning and evaluation in an open system does not solely rely on data such as reputation of users in its own user community, but also take advantage of diverse information sources in the Internet, such as from other systems: social networks, community platforms, recommender systems, content sharing systems, or search engines. By adopting an open architecture, reputation data and related relevant information that are shared from one system to another can be used as prior knowledge to learn user behaviors across systems. The problem of "cold start" when estimating trust worthiness of a newly joined participant can be mitigated.
2. Regarding the management of user identities, an open trust system does not exclusively rely on system-internal identity but also on Web-based entities, which are digital footprints left by users on the Web and therefore not easily manipulated. Also, by employing appropriate data mining techniques, an open trust system has much better potential to counter identity attacks, which are a major problem in current closed systems. Therefore, aggregating reputation information from different contexts is expected to more effective in discovering collusive groups and in dealing more effectively with "Sybil" attacks (Douceur, 2002). An open trust system is therefore more resilience to whitewashing behavior than a system working only with its predefined identity management mechanism.
3. In dynamically formed emerging communities, self-organizing community processes form an agreement on the instantiation of the computational trust model for a given context. These processes involve identifying and structuring sources of reputation data, extracting and agreeing on behaviors of interest (in particular specify malicious behavior) and their

evaluation, and establishing shared evaluation metrics. The obvious source for building trust and reputation metrics is relevance feedback from participants to determine categories of behavior and their evaluation. In this process, a community norm is compared and adapted based on the norms of other communities, such as existing recommender systems, emergent semantic systems and web-ranking systems, and it can identify subcommunities with shared interests. To this end, a trust evaluation model cannot be applied in these communities a priori, without hindering their formation. An open system can inherit and address trust-related issues based on the norms on expectable behavior and community assumptions from the other similar systems in the same application domain. Shared but hidden assumptions of communities on values and evaluations of actions can be discovered by aggregation ratings and other reputation information across communities that otherwise would be hard to discover and specify.

4. Beneficial attributes of each computational trust model employed by different systems on different communities can be combined together. Such combination of different models, with availability of more data from many systems help the trust learning and evaluation process becomes more resilient against various attacks from users with malicious behaviors. This possibility will be analyzed in detail in later sections of this chapter.

6.3 Related Work

One of the first initiatives (2008) to build an open reputation systems is that of the OASIS standardization forum. Specifically, an Open Reputation Management Systems (ORMS) technical committee[1] has been founded to propose appropriate standards for representing reputation data and standard definitions of reputation scores across different systems. The standards will provide the means for understanding the relevance of a rating score within a given transaction across communities. However, the means of exchanging and combining computational trust models or exploiting the identity management mechanisms among different systems have not been considered.

(Pingel & Steinbrecher, 2008) introduces a software prototype to store ratings of users participating in different online-discussion fora. This work assumes the complete trustworthiness of the identity provider and the users have full-control on the disclosure of actions and reputation information across communities.

The building of open reputation systems is also well-related to existing works on federated identity management (Bhatti *et al.*, 2007). For example, (Gutscher, 2007) defines a PKI framework for secure referral dissemination across communities. (Rehák & Pechoucek, 2007) proposes a generalized approach to model contextual environment of an agent and compute its trust values in a new context based on distance with the related situations. Emerging standards such as OpenID[2] and OpenAuth[3] enable the creation of portable identities and allowing community

[1] http://www.oasis-open.org/committees/orms

[2] www.openid.net

[3] www.openauth.net

managers to delegating access to users across system boundaries.

The issues of knowledge representation and information sharing across different systems also attract much research efforts. For example, the use of Semantic Web RDF models as FOAF (Friend Of A Friend)[1] to weave online social networks, as the SIOC initiative (Semantically-Interlinked Online Communities)[2] to weave social activities such as blog comments could provide a complete interlinked graph of user profiles on top of existing applications. Also, different community semantics can be expressed in meta-models such as OWL[3]. However, the adoption of common ontologies or schemas is still relatively slow. Moreover, the process of community building is highly dynamic. Emergent communities may dynamically form within existing ones with different interactions and objectives. As explained in (Aberer *et al.*, 2004), semantic interoperability should be viewed as an emergent phenomenon constructed incrementally, and its state at any given point in time depends on the frequency, the quality and the efficiency with which negotiations, such as those defined in (Aberer & Hauswirth, 2003; Aberer *et al.*, 2003b), can be conducted to reach agreements on common interpretations within the context of a given task. Specifically, in (Aberer & Hauswirth, 2003), semantic interoperability is addressed by means of local schema agreements between data sources through queries and gossiping of schema mappings. The quality of semantic mappings is gradually improved by a feedback algorithm until to finally achieve global agreement selecting the right data sources for schema translation.

Commercial solutions for aggregating online reputation information related to a person are also available and becoming increasingly popular, such as Online Reputation Monitor[4], Reputation Manager[5], or Reputation Defender[6]. However, these are mainly entity matching Web search engines (Shen *et al.*, 2005a) that neither automatically check the accuracy of information nor aggregate it into a single trustworthiness metric.

To the best of my knowledge, this work is the first attempt to systematically study and analyze the benefits and challenges of developing open trust management systems. It is also the first work on studying benefits of information sharing among across trust management systems and opportunities of combining different computational trust models to enhance the system capabilities in detecting misbehavior.

6.4 Reference Architecture

Based on the common working principles of reputation-based trust management systems available in existing applications, a reference architecture for an open trust system can be developed such as in Fig. 6.1.

A traditional (reputation-based) trust management system, as shown in the left most of the

[1] www.foaf-project.org

[2] www.sioc-project.org

[3] http://www.w3c.org/TR/owl-ref/

[4] http://reputation.distilled.co.uk/

[5] www.reputationmanager.com

[6] www.reputationdefender.com

architecture, include various components for the modeling, learning, computation of trust, as well as for the collection of trust-related data, the conventional (relational) database systems for storage and processing of data related to user actions and activities.

An open trust management system extends the traditional architecture with three additional components: a knowledge acquisition, a knowledge sharing, and a data adaptation layer. The data adaptation layer acts as an intermediate component for transforming data into standardized concepts to share to other systems, and to transform acquired knowledge from other systems to be used by the legacy system.

The knowledge acquisition layer contains various modules to enable an open system to collect various types of data from other systems on the Web. This information may include data related to credentials and activities of users who participate in this system and also actively participates in other systems under different identities. Also, knowledge such as behaviors of malicious uses, their attack strategies, and corresponding counter-measures on other similar trust management systems can be collected and integrated into the current system. The computational trust model of an open system should also be able to integrate these collected information and perform estimation on trustworthiness of participants accordingly. Furthermore, it can be inferred that data to be used by an open trust system needs not be stored locally but can be from various sources on the Web (hence for data storage and processing in an open system a Web DBMS might be better suitable). Due to privacy issues, these sources may only reveal anonymized or encrypted version of the necessary data to the current system when being required and thus the computational trust model needs to operate on such data in a privacy-preserving way.

The knowledge sharing layer offers data and information of the current system to other via appropriate services. Information to be shared may include user identification service (identity resolution service), reputation and trust-relation information on the user (reputation data sharing), known adversarial models and corresponding solutions, as well as the computational trust model service to other systems.

An open trust system may integrate with similar systems at different levels, such as depicted in Fig. 6.2. The key advantage of an open trust system is that its capabilities can be effectively combined with that of different open systems via the exporting of user identities, reputation data, and attack models as a service. Such a combination, however, relies on the semantic interoperability among the systems. That is, how an open system interprets semantics of information and services provided by the other systems, e.g., the meaning of transactions, ratings, reputation, and trust on users. Again, due to security and privacy reasons, privacy-preserving methods of data integration should be applied in this knowledge sharing process.

6.5 Challenges in Developing Open Trust Systems

The possible benefits of building open reputation-based trust systems are not without significant challenges. This section will identify and discuss several of these most important issues. These questions are presented in order of relevance and importance in my opinion, most of which however are inter-related. Any solution to a specific question should be developed giving

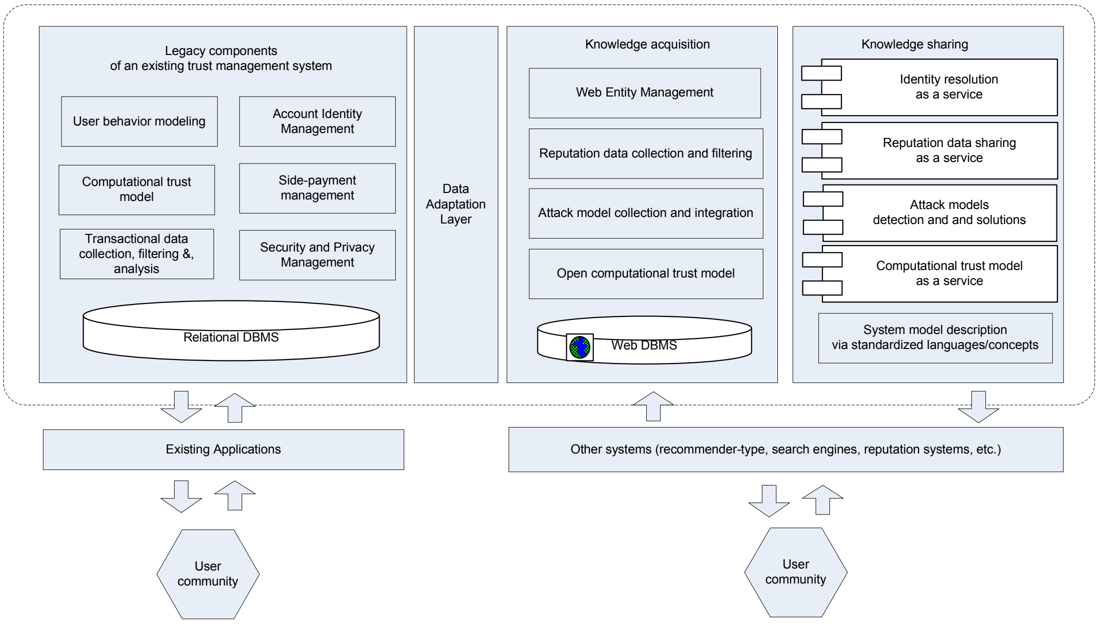

Legacy components
of an existing trust management system
User behavior modeling
Account Identity Management
Computational trust model
Side-payment management
Transactional data collection, filtering &, analysis
Security and Privacy Management
Relational DBMS
Data Adaptation Layer
Knowledge acquisition
Web Entity Management
Reputation data collection and filtering
Attack model collection and integration
Open computational trust model
Web DBMS
Knowledge sharing
Identity resolution as a service
Reputation data sharing as a service
Attack models detection and and solutions
Computational trust model as a service
System model description via standardized languages/concepts
Existing Applications
Other systems (recommender-type, search engines, reputation systems, etc.)
User community
User community

Figure 6.1: Architecture of an open reputation-based trust management system

consideration of the other issues.

6.5.1 Opening systems for knowledge sharing

The building of open trust systems assumes that a system may acquires reliable and semantically interpretable data, including user identity and reputation data from the other systems, namely community portals, search engines, social networks, and other (open or closed) reputation systems. The opening up existing (reputation) systems and combing them for synergic benefits are not without fundamental challenges.

Incentives of sharing knowledge: Aggregating reputation within a community is a costly process. Furthermore, reputation information can be considered as a feature of the community and a public good for the community members. Therefore, it is not straightforward that community managers have sufficient *incentives to disclose reputation information or any knowledge to other communities*, especially when such information may goes competing systems. For example, an existing e-trading system with very large userbase, namely *eBay.com* is unlikely to be willing share reputation data with another smaller system, such as *ricardo.ch*.

Usually, online communities do not provide any inside information to outsiders for free. The reason is that the disclosure of information on identities and their relations are of high importance. Even raw transaction data could also be useful as various data mining can be used to discover various user preferences and correlation among them. Besides their intended usage for anomaly and misbehavior detection, these discovery may also be valuable for other purposes.

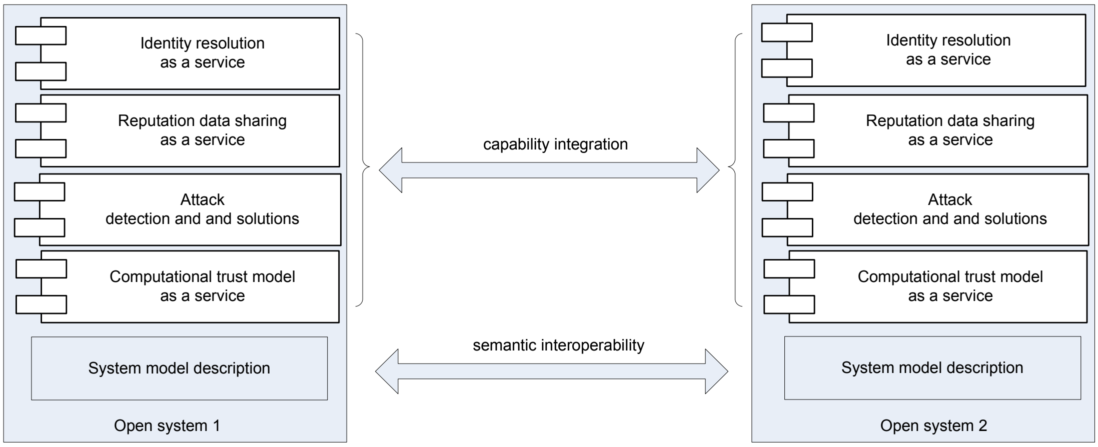

Identity resolution as a service
Reputation data sharing as a service
Attack detection and and solutions
Computational trust model as a service
System model description
Open system 1
capability integration
semantic interoperability
Identity resolution as a service
Reputation data sharing as a service
Attack detection and and solutions
Computational trust model as a service
System model description
Open system 2

Figure 6.2: Different levels of knowledge sharing among open reputation-based trust systems

For example, Facebook's user profile data are valuable information to market researchers and thus can not be given for free by Facebook.

Peering agreements among communities to share reputation-related information could create *externalities* for both communities by increasing their credibility to the users. This scenario particularly makes sense for federated small communities that are thus able to calculate trustworthiness more effectively over a larger user and transaction base and alleviate the "cold start" problem. Therefore, *smaller communities are expected to be more willing to reciprocate*. On the other hand, large well-established communities are not expected to be willing to disclose reputation-related data for free. Economic incentives can be provided to them by means of side-payments to provide reputation, identity or raw transaction data. However, in that case, competition among reputation-related information providers would arise. The accuracy of reputation or identity information, the size of the community, the number of competing providers and the number of clients would determine actual prices. A federated group of many small communities can eventually be a strong reciprocative partner for a large system. For example, *eBay* may eventually lose members if ricardo.ch exchanges data with *eBay*'s competitor, such as *alibaba.com*. Consequently, community manager of systems of different scales may only be willing to share data if the data exchange is fair and equivalently benefit to them. Automatic protocols for *privacy-preserving* and *fair data exchange* among different systems are thus highly required.

Reliability of shared data: An inherent problem of each reputation system are unreliable (biased) ratings posted by users with different motives. A similar problem emerges when such knowledge is shared between different communities. Community managers may have the incentive to improperly manipulate reputation information. To this end, a trustworthiness metric should be attached to each community regarding the accuracy of reputation or any information reported. Also, incentives for accurate reputation information exchange have to be provided, i.e. exchange on a reciprocative basis or buying/selling reputation information (Tadelis, 1999).

User privacy and reputation protection: The Internet exposes personal information without providing any incentive for accountability. This information is subject to aggregation on-demand by various commercial systems. For example, systems such as Spokeo.com make deal with several systems, search and provide almost a lot of information such as photos, blog entries, visits to different sites, of any people knowing only his or her nickname in one online communities. These systems for aggregating such personal information may thus (accidentally) use unreliable information sources and even fake data, which may severely affect the reputation of a person (hence defamation attack). However, the aggregation of even accurate reputation information and its linkage to a person may raise privacy concerns for the members of online communities. Given such an observation, *sharing user data across systems may deter participation level of privacy-aware users*. Community managers thus have another reason not to share their data to keep their system growth steadily.

To mitigate the above problem, the sharing of information and knowledge shall ensure that individual actions and interactions inside a community should not be linkable to real identities, but only to pseudonyms. Moreover, only relevant reputation information should be communicated across communities. Appropriate techniques, such as cryptography and obfuscation should be employed to *protect user data and minimize information leakage*. In addition, improving reliability of information sources may be a substantial aid in fighting malicious publication of personal information, defamation attacks and contribute to better protection of personal privacy and integrity.

6.5.2 Knowledge acquisition and system integration

An apparently challenging issue in the knowledge acquisition of an open trust system is to address the semantic interoperability among different systems. Even with a certain level of semantic agreement, for example, the case of sharing information and knowledge from two systems on the same application domain, data integration is also non-trivial. Apart from these challenges, there are also several other issues relevant to the knowledge acquisition process that need to be address while building any open trust system.

Which knowledge to be shared and acquired: The extent to which data to be shared between systems may well depend on the openness of each system. Possible levels of information sharing include: sharing user identities and reputation data, sharing computational trust models, sharing knowledge regarding of user behavioral patterns and malicious activities, vulnerabilities, adversarial attacks and corresponding detection and appropriate counter measures. It is expected that with more shared and acquired knowledge, the performance of the system shall improve. However, *quantifying measurements of such improvements given the knowledge level acquired* in practical applications are not easy and deserve further studies.

It is also a question whether it is best to *integrating raw transaction data or reputation information* from another system. When integrating ratings and trust values rather than raw transaction data, the underlying semantics of generating these rating have to be taken into account. It is interesting that, depending on the trust evaluation model employed, the same reputation data can lead to quite diverse evaluations (Despotovic & Aberer, 2006). Furthermore, the dynamics and the structure of a certain community have to taken into account, e.g. kind of transactions, frequency, etc. Also, the credibility of feedback for raw transactions and the reliability assigned to a community for the reputation information provided is very important.

Identity resolution issues: One of the most critical short-comings of closed reputation-based trust systems is the possibility of whitewashing behaviors by acquiring cheap identities. In an open trust system, the virtual identities of the participants can be related to existing contexts in a variety of other systems where the users may present, such as social networks or Semantic Web entities. However, an emerging question is *how to resolve identities when integrating reputation systems from different communities*.

An appropriate solution to the above fundamental question requires the combination of different techniques to detect of digital footprints of users across communities. First, certain online communities may employ membership based on real-life attribute credentials. E.g., *ricardo.ch* is an auction site that uses physical addresses as a means of verification of user identities. Some online social networks may even employ real identities or real life addressing for membership registration. Second, users may employ the same credentials across several communities and may even exhibit similar behaviors. Thus indirect identity resolution techniques by analyzing correlated patterns of actions across different contexts to disambiguate the identity of participants can be well applicable. Besides, the Semantic Web is increasingly providing more ways to manage entities, including users, more easily and systematically, by relating information from widely spread resources to a subject of trust evaluation via Friend-Of-A-Friend networks, (Shen *et al.*, 2005a). Such entity traits, credentials and behavioral patterns of users on the Web constituent their digital footprints, which cannot be easily created or changed. For example, an indirect way for identity resolution would be to combine information from different trust and rating graphs. This is analogous to the use of social network analysis. It has been proved in (Backstrom *et al.*, 2007; Narayanan & Shmatikov, 2009) that if only the a limited number of links are known in an anonymous social network, it is possible to discover the true identities of all other nodes of the network. Also, employing graph relations from different communities and complex network metrics (e.g. clique), it is possible to discover collusive groups and deal more effectively with "Sybil" attacks. In short, *discovering digital footprints of users and building an effective identity management mechanism* based on these footprints are challenging yet promising issues in the implementation of an open trust management model.

Accuracy of identification and incentives of cooperation: The resolution of user identities among different contexts may be inaccurate. In this case, reputation and trust graph information is improperly mapped into the integrated system, which introduces some noise in estimation of trustworthiness of its users. Therefore, the incentives to users to exhibit an acceptable behavior may get distorted. The *relation between the accuracy of the identity resolution and the resulting incentives for users* after aggregation of reputation information from different communities has to be investigated. Also, the aggregation of reputation information across communities and the subsequent trust evaluations also influence the dynamics inside communities. For example, a member evaluated as untrustworthy by other communities will be regarded as less trustworthy inside a community where it behaves cooperatively. As a result, the member's dominant strategy in the latter community is not to cooperate. It thus requires much more effort from the particular member than others to be regarded as trustworthy in an emerging community.

Reputation semantic, community structure and system dynamics: Reputation within a certain context is calculated based on the specific metrics of reputation within that context (i.e. how is good reputation defined, how is rated), the specific computational trust model in use (Vu & Aberer, 2008b), the community structure and dynamics (i.e. interactions), the community norms on what is good or bad behavior (i.e. reputation semantics) and evaluation criteria. Past transactions may be rated or not, by one or both transacted parties, and with quantitative or qualitative feedback. The feedback messages are aggregated based on different computational trust models that may also weight the significance of the ratings by the credibility of the raters and take into account other factors, such as the transaction context factor (i.e. size and type of each transaction) and the community context factor (e.g. common incentives or beliefs) (Xiong & Liu, 2004). Rating, reputation, credibility metrics and the metrics of any other factor for reputation calculation and their semantics may have to be shared across different communities for reputation transferability.

To this end, OASIS standards[1] will provide the means for representing and understanding the relevance of a reputation score across communities, but not the algorithms for computing the scores. However, the context of a community also has to be effectively modeled and described. The semantic distance of the contexts has to be calculated in terms of relativeness or usefulness (as opposed to lexicographical, linguistic or physical) distance between their attributes (Roddick *et al.*, 2003).

Different communities may have different norms on which behavior is acceptable or not. Therefore, a physical entity with consistent behavior across communities may have high reputation in one community and low reputation in another. The evaluation criteria are also affected by the community norms but also from the context structure and community assumptions. Also, different communities have different assumptions on the way they derive trust from reputation and activities of the users. As a result, even the same physical entity, being member of different communities may act differently. For example, consider a member of eBay that is reluctant of giving negative feedback fearing provider's retaliation, while in Amazon, the same person is very strict in her evaluation and mostly provides negative book reviews. Therefore, the assumptions and the structure of each community have to be taken into account when aggregating reputation from different contexts. A *knowledge representation language that is sufficiently expressive and possibly light-weight* for the description of the reputation semantics, its context, context structure and the related assumptions is strongly required.

How to best combine different computational trust models: An open trust system may employ different computational trust models provided by several similar systems to detect misbehavior. Furthermore, the system is also provided with more data via appropriate knowledge sharing and acquisition process.

[1] http://www.oasis-open.org/committees/orms

As each individual system may employ a specialized computational trust model that is most robust against some certain adversarial attack, it is expected that the open trust system that uses a combination of different trust models would outperforms existing closed systems in various aspects. First, the open system should be resilient to almost every attacks as each individual system that provides computational models and reputation-related data for the system. *The open system shall also be able to detect various misbehavior patterns more than each individual system does.*

Given the above expectation, the most interesting research question is how to combine different computational trust models provided by other systems in the most effective way. A potential approach to effectively address this question is to use weighting and boosting techniques in machine learning literature (Freund & Schapire, 1995). Such techniques help to build an open computational trust model that is provably much more robust than individual models under any malicious behaviors. Such an improvement in detection accuracy of the trust models also implies increased incentives for cooperation in the system (Vu & Aberer, 200x).

Integration cost vs. benefits: Given the aforementioned issues, aggregation of reputation information across different communities requires identity resolution, collection of reputation information and trustworthiness estimation and thus is potentially costly.

Therefore, *one should consider various integration cost and assess potential benefits of different levels of integration*, for example, which data to be shared and in what ways, when implementing such an open trust system. For example, as it involves considerable communication and processing latency, it is not considered as feasible for an (open) trust system to use knowledge and information from other systems for its trust evaluation on demand. Also, it is very costly in terms of communication to continuously exchange reputation information with other communities for every user or to continuously run crawlers for this purpose. A hybrid approach combining proactive crawling over other communities for the reputation information of selected members of the community and on-demand reputation information requests for some members to other communities seems to be more appropriate.

6.6 Naive Combination of Computational Trust Models

This section presents a generic model that evaluates the benefits of combining the computation trust models of two similar systems. Herein, we consider the two reputation systems being used in a similar application contexts. Therefore, ratings and reputation values in one system are well-defined and relevant (i.e. meaningful) to the other. An example of two such systems are the online e-commerce sites eBay.com and ePier.com, which employ similar reputation mechanisms.

My goal is to evaluate whether such a combination of two reputation systems leads to any improvement of the robustness of each individual system against user misbehavior. Specifically, the resilience of the combination of two trust models, each of which is designed to be resilient against a certain misbehavior model will be measured and compared to the resilience of individual models. The goal is to quantify the improvement (in terms of overall resilience) of the combined trust model as compared to each individual and to measure the synergy effect of such a combination, if any.

It is assumed that community managers and initiators provide basic support for identifying and structuring relevant reputation data for mapping it into a representation with shared semantics. Therefore, exchanges of reputation data and combination of information from different systems can be done effectively. For simplicity, each community manager is assumed to be trustworthy.

A reputation-based trust system can be formally described as in Def. 16. As defined therein, only *single-dimensional* reputation values with *common* and *well-defined* semantics are considered.

Definition 16 *A reputation-based trust management system* $\mathcal{R}$ *is a 6-tuple* $\mathcal{R} = \langle \mathcal{U}, id, \mathcal{K}, \mathcal{H}, \mathcal{T}, \mathcal{A} \rangle$ *where:*

- $\mathcal{U}$ *is the set of usernames in the system. This information is usually public.*
- $\mathcal{K}$ *is the set of credentials to verify user identities. In most cases* $\mathcal{K}$ *is private, i.e., only known by the owning user and the community manager. Example credentials are emails, public keys, or even physical addresses.*
- $id : \mathcal{U} \rightarrow \mathcal{K}$ *is the system identity verification method (which may be done during the registration phase).*
- $\mathcal{H} = \{h^u : u \in \mathcal{U}\}$ *is historical performance data of all users.*
- $\mathcal{T}$ *is the set of completed transactions among users.*
- $\mathcal{A} : \mathcal{U} \times \mathcal{T} \times \mathcal{H} \rightarrow \{trustworthy, undecided\}$ *is an algorithm operating on* $\mathcal{H}$ *and outputs a value determining if a user* $u \in \mathcal{U}$ *is trustworthy for a given transaction* $t \in \mathcal{T}$.

Let $\mathcal{B} = \{b^u, u \in \mathcal{U}\}$ be the overall behavior model of all users, where b^u is the behavior of a user u on the probability space. We will make no assumption on each individual behavior b^u, thus the overall model $\mathcal{B}$ summarizes any possible malicious and opportunistic behaviors of any participants, including any collusive actions of a group.

The misclassification error rate of a reputation system $\mathcal{R} = \langle \mathcal{U}, id, \mathcal{K}, \mathcal{H}, \mathcal{T}, \mathcal{A} \rangle$ given a user behavior model $\mathcal{B}$ is defined as the expectation that the system misclassifies behaviors of any user for a transaction (Def. 17). A lower misclassification error rate implies a higher resilience under malicious behaviors of users.

Definition 17 *Let* $0 \leq e(u, t \mid \mathcal{A}, \mathcal{B}, h^u) \leq 1$ *be the probability the algorithm* $\mathcal{A}$ *misclassifies a user* u *as trustworthy for a transaction* $t \in \mathcal{T}$*, given the rating history of the user* h^u *and the overall behaviors of all users* $\mathcal{B}$*. The misclassification error rate of the system given the user behavior model* $\mathcal{B}$ *is* $s = \frac{\sum_{u \in \mathcal{U}, t \in \mathcal{T}} e(u,t|\mathcal{A},\mathcal{B},h^u)}{|\mathcal{U}||\mathcal{T}|}$*. The system resilience under the behavior model* $\mathcal{B}$ *is* $r = 1 - s$.

Denote $h^u \parallel \Delta^u$ the integration of two historical performance data sets h^u and Δ^u of a user u. In this work, the analysis is limited to those algorithms $\mathcal{A}$ satisfying the following Hypothesis 1. Such a hypothesis is practically reasonable with a wide range of algorithms, since a better learning result is usually expected given more data.

Hypothesis 1 *$e(u, t \mid \mathcal{A}, \mathcal{B}, h^u)$ is a monotonically decreasing function of $\mid h^u \mid$. In other words, $d(\Delta^u, \mathcal{A}, \mathcal{B}) = e(u, t \mid \mathcal{A}, \mathcal{B}, h^u \parallel \Delta^u) - e(u, t \mid \mathcal{A}, \mathcal{B}, h^u) \leq 0$*

Let us combine two systems $\mathcal{R}_1$ and $\mathcal{R}_2$ with the same identity credential set $\mathcal{K}$. Denote $\mathcal{R}_i = \langle \mathcal{U}_i, id_i, \mathcal{K}, \mathcal{H}_i, \mathcal{T}_i, \mathcal{A}_i \rangle$, where $\mathcal{H}_i = \{h_i^u : u \in \mathcal{U}_i\}$, $i = 1, 2$, our goal is to combine $\mathcal{R}_1$ and $\mathcal{R}_2$ in order to achieve an overall resilience that is better or, at least, not worse than, each individual system under any misbehavior model $\mathcal{B}_j, j = 1, 2$. The integration of more than two systems can be done similarly.

For example, let $\mathcal{A}_1$ be a dishonesty detection algorithm designed to discover collusive groups of malicious users who consistently vote in favor for each other (possibly for a limited group size). Let $\mathcal{A}_2$ be another algorithm that detects periodical changes in behaviors of users, where period is finite. We will investigate if by sharing information and combining the two systems $\mathcal{R}_1$ and $\mathcal{R}_2$ together, the final system is able to detect both collusively malicious and periodically changes of behaviors more accurately. Moreover, we want the resilience of the integrated system to be higher than the resilience of each individual system.

I will show that such synergies can be achieved by combing the knowledge of two systems. Particularly, I exploit the fact that many users participate in both systems and their behaviors can be derived by aggregating their historical performance statistics in both systems. The detection of common participants of the two systems is related to the problems of alias detection and entity resolution in security and database research communities. For example, in (Narayanan & Shmatikov, 2009), the authors managed to identify a large number of user identities from anonymized Flickr and Twitter data sets by analyzing their relationships with others with a very high accuracy, even if the two data sets are not strongly overlapped.

Basically, such an *identity equivalence mapping* can be done based on many information sources: First, many users use the same credentials in both system that uniquely identify them from the others (a.k.a. their digital footprints). For example, a user may use the same e-mail address to register as participants in the two systems. Other typical credentials can be detected from similarity in IP source addresses, correlations in posting timestamps, etc. An example solution to automatically detect such identity equivalences from various information sources is presented in (Cudré-Mauroux *et al.*, 2009). Second, even if users use different credentials, equivalence among their usernames in two systems can be derived by analyzing the (trust) relationships among these users with the others. For example, methods to look for matching hidden patterns and structural stenography among users in the trust graphs of two systems (Backstrom *et al.*, 2007; Narayanan & Shmatikov, 2009) can be applicable. Figure 6.3 illustrates the possibilities of such an identity equivalence mapping. Suppose three users u, v, w forms a collusion group in the first system $\mathcal{R}_1$, and other users u', x, y form another collusion in $\mathcal{R}_2$. By combining the two systems, algorithm $\mathcal{A}_1$ can be used in both communities to detect the two collusive groups effectively. Since u and u' actually use the same credential, e.g., the same email for identity verification, they are revealed to be associated to the same user. We can then estimate that v, w is related to x, y with certain probability.

In this work, specific solutions to this problem of identity resolution will not be investigated. We will assume the availability of an appropriate identity equivalence mapping mechanism for

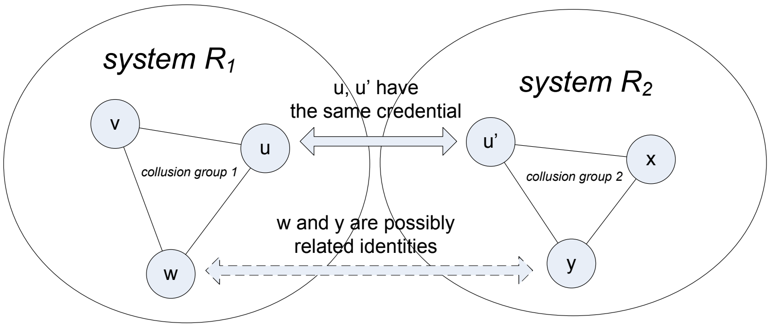

system R_1
system R_2
u, u' have the same credential
v
u
w
collusion group 1
u'
x
y
collusion group 2
w and y are possibly related identities

Figure 6.3: Inferring identity equivalence in the two reputation systems based on known knowledge and relationship analysis.

the integration of the two reputation systems. We define as $\tau = \{u \in \mathcal{U}_1 : \exists v \in \mathcal{U}_2, id_1(u) = id_2(v)\}$ the set of usernames with the same identity credential in both systems. Similarly we can define $\tau' = \{u \in \mathcal{U}_2 : \exists v \in \mathcal{U}_1, id_1(u) = id_2(v)\}$, and $\mid \tau \mid = \mid \tau' \mid$. Def. 18 formally introduces the notion of identity equivalence mapping between two user communities.

Definition 18 *An identity equivalence mapping between the user communities of two systems $\mathcal{R}_1$ and $\mathcal{R}_2$ is defined as:*

$$ide : \mathcal{U}_1 \times \mathcal{U}_2 \times \tau \longrightarrow [0, 1] \tag{6.1}$$

The function $ide(u, v, \tau)$ gives the probability that user $u \in \mathcal{U}_1$ is the same as user $v \in \mathcal{U}_2$, knowing the relationship among users and the initial set of equivalent identities τ.

We denote as τ^* the set of users that can be reliably detected as common in both system by identity equivalence mapping given τ. Formally, $\tau^* = \{u \in \mathcal{U}_1 : \exists v \in \mathcal{U}_2, ide(u, v, \tau) = 1\}$, and presumably $\tau^* \supseteq \tau$.

Similarly, we have $\tau'^* = \{u \in \mathcal{U}_2 : \exists v \in \mathcal{U}_1, ide(u, v, \tau') = 1\}$, where $\tau'^* \supseteq \tau'$, and $\mid \tau^* \mid = \mid \tau'^* \mid$ (see Figure 6.4).

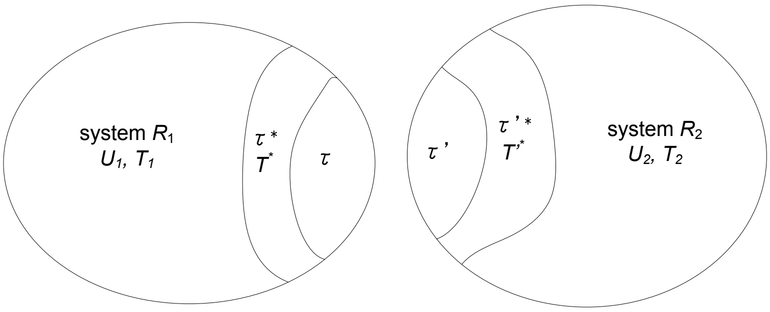


Figure 6.4: Combination of two reputation systems

Given the above notations, we introduce and analyze the resilience of the following naïve combination of the two reputation systems (Def. 19).

Definition 19 *The naïve combination of two reputation systems $\mathcal{R}_i, i = 1, 2$ is a reputation system $\mathcal{R} = \langle \mathcal{U}, id, \mathcal{K}, \mathcal{H}, \mathcal{T}, \mathcal{A}\rangle$ defined by:*

- $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2 - \tau^*$.
- $\mathcal{H} = \mathcal{H}_1 \parallel \mathcal{H}_2$ *is the combination of historical data from $\mathcal{R}_1$ and $\mathcal{R}_2$. Specifically:* $\mathcal{H} = \{h_1^u \parallel h_2^u : u \in \tau^*\} \cup \{h_1^u : u \in \mathcal{U}_1 - \tau^*\} \cup \{h_2^u : u \in \mathcal{U}_2 - \tau'^*\}$.
- $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$
- $\mathcal{A} = \mathcal{A}_1 \circ \mathcal{A}_2$ *is the combination of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. Under $\mathcal{A}$, a user u is evaluated as trustworthy iff given the same history of u, each $\mathcal{A}_1$ and $\mathcal{A}_2$ independently evaluates u as trustworthy.*

Let $s_{ij} = \frac{\sum_{u\in\mathcal{U}_i, t\in T_i} e(u,t|\mathcal{A}_i,\mathcal{B}_j,h_i^u)}{|\mathcal{U}_i||\mathcal{T}_i|}$, be the misclassification error rate of $\mathcal{R}_i$ given that the user behavior is $\mathcal{B}_j$, where $i, j \in \{1, 2\}$. Suppose that $\mathcal{A}_i$ is designed to be more robust against the behavior model $\mathcal{B}_i$ than to the model $B_j, j \neq i$, it follows that $s_{ii} \leq \min\{s_{ij}, i \neq j\}$, where $i, j \in \{1, 2\}$.

Denote $\overline{s}_i = \frac{\sum_{u\in\mathcal{U}, t\in\mathcal{T}} e(u,t|\mathcal{A}_1\circ\mathcal{A}_2,\mathcal{B}_i,h_1^u\|h_2^u)}{|\mathcal{U}||\mathcal{T}|}$ the misclassification error rate of the naïve combination of two systems (Def. 19) given the user behavior model $\mathcal{B}_i, i = 1, 2$. Proposition 5 gives us the estimation of the resilience of the combined system.

Proposition 5 *Let $\mid \mathcal{U}_2 \mid = \alpha \mid \mathcal{U}_1 \mid$, $\mid \mathcal{T}_2 \mid = \beta \mid \mathcal{T}_1 \mid$, and $\mid \tau^* \mid = \gamma \mid \mathcal{U}_1 \mid$, where $0 \leq \alpha, \beta, \gamma \leq 1$. An upper-bound of the worst-case misclassification error of the combined system is given by:*

(a) $\overline{s}_1 \leq \frac{s_{11}+\alpha\beta s_{21}}{(1+\alpha-\gamma)(1+\beta)}$.

(b) $\overline{s}_2 \leq \frac{s_{22}+\alpha\beta s_{12}}{(1+\alpha-\gamma)(1+\beta)}$.

Proof 10 *Due to symmetry, we only need to prove (a). We note that under any user behavior model $\mathcal{B}_j, j = 1, 2$, Def. 17 and Def. 19 give us:*

$$e(u, t \mid \mathcal{A}_1 \circ \mathcal{A}_2, \mathcal{B}_j, h^u) = e(u, t \mid \mathcal{A}_1, \mathcal{B}_j, h^u)e(u, t \mid \mathcal{A}_2, \mathcal{B}_j, h^u) \tag{6.2}$$

From Hypothesis 1:

$$d(\delta^u, \mathcal{A}_i, \mathcal{B}_j) = e(u, t \mid \mathcal{A}_i, \mathcal{B}_j, h_k^u \parallel \delta^u) - e(u, t \mid \mathcal{A}_i, \mathcal{B}_j, h_k^u) \leq 0 \tag{6.3}$$

where $i, j, k \in \{1, 2\}$.

Let $\mathcal{T}^* \subseteq \mathcal{T}_1$ (resp. $\mathcal{T}'^* \subseteq \mathcal{T}_2$) be the set of transactions where estimates of behavior of users in τ^* (resp. τ'^*) are made, the total misclassification error TME by the combined systems is given by:

$$\begin{aligned} TME \triangleq \sum_{u\in\mathcal{U}, t\in\mathcal{T}} e(u, t \mid \mathcal{A}_1 \circ \mathcal{A}_2, \mathcal{B}_1, h_1^u \parallel h_2^u) \quad &= \\ \sum_{u\in\tau^*, t\in\mathcal{T}^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_1^u \parallel h_2^u)e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_1^u \parallel h_2^u) \quad &+ \\ \sum_{u\in\mathcal{U}_1-\tau^*, t\in\mathcal{T}_1-\mathcal{T}^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_1^u)e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_1^u) \quad &+ \\ \sum_{u\in\tau'^*, t\in\mathcal{T}'^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_1^u \parallel h_2^u)e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_1^u \parallel h_2^u) \quad &+ \\ \sum_{u\in\mathcal{U}_2-\tau'^*, t\in\mathcal{T}_2-\mathcal{T}'^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_2^u)e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_2^u) \quad & \end{aligned} \tag{6.4}$$

On the other hand, considering the two trust management systems separately, the sum SME of their misclassification errors is given by:

$$\begin{aligned} SME &\triangleq \sum_{u \in \tau^*, t \in \mathcal{T}^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_1^u) \\ &+ \sum_{u \in \mathcal{U}_1 - \tau^*, t \in \mathcal{T}_1 - \mathcal{T}^*} e(u, t \mid \mathcal{A}_1, \mathcal{B}_1, h_1^u) \\ &+ \sum_{u \in \tau'^*, t \in \mathcal{T}'^*} e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_2^u) \\ &+ \sum_{u \in \mathcal{U}_2 - \tau'^*, t \in \mathcal{T}_2 - \mathcal{T}'^*} e(u, t \mid \mathcal{A}_2, \mathcal{B}_1, h_2^u) \end{aligned} \tag{6.5}$$

Since $0 \leq e(.) \leq 1$*, each term in Eq. (6.4) is less than the corresponding term in Eq. (6.5). Using Eqs. (6.2),(6.3), we then have:*

$$\begin{aligned} TME - SME &\leq \sum_{u \in \tau^*, t \in \mathcal{T}^*} d(h_2^u, \mathcal{A}_1, \mathcal{B}_1) && (6.6) \\ &+ \sum_{u \in \mathcal{U}_1 - \tau^*, t \in \mathcal{T}_1 - \mathcal{T}^*} 0 \\ &+ \sum_{u \in \tau'^*, t \in \mathcal{T}'^*} d(h_1^u, \mathcal{A}_2, \mathcal{B}_1) \\ &+ \sum_{u \in \mathcal{U}_2 - \tau'^*, t \in \mathcal{T}_2 - \mathcal{T}'^*} 0 \\ &= \sum_{u \in \tau^*, t \in \mathcal{T}^*} d(h_2^u, \mathcal{A}_1, \mathcal{B}_1) \\ &+ \sum_{u \in \tau'^*, t \in \mathcal{T}'^*} d(h_1^u, \mathcal{A}_2, \mathcal{B}_1) \\ &\triangleq \Phi(\tau^*, \mathcal{T}^*, \tau'^*, \mathcal{T}'^*) \leq 0 && (6.7) \end{aligned}$$

The left-hand size of (6.4) can be rewritten as:

$$TME = (\mid \mathcal{U}_1 \mid + \mid \mathcal{U}_2 \mid - \mid \tau^* \mid)(\mid \mathcal{T}_1 \mid + \mid \mathcal{T}_2 \mid)\overline{s}_1 \tag{6.8}$$

We also have:

$$\begin{aligned} SME &= \mid \mathcal{U}_1 \mid\mid \mathcal{T}_1 \mid s_{11} + \mid \mathcal{U}_2 \mid\mid \mathcal{T}_2 \mid s_{21} \\ &= \mid \mathcal{U}_1 \mid\mid \mathcal{T}_1 \mid (s_{11} + \alpha\beta s_{21}) \end{aligned}$$

Thus (a) follows naturally.

Suppose $s_{ii} > 0$ and let $s_{ji} = \delta_i s_{ii}, i, j \in \{1, 2\}, j \neq i$, where $\delta_i \geq 1$. Given Proposition 5, it is apparent that the combination of the two systems results in a better system if and only if $\overline{s}_i \leq s_{ii}, i = 1, 2$. This is equivalent to the following condition:

$$f_i = \frac{(1 + \alpha - \gamma)(1 + \beta)}{1 + \alpha\beta\delta_i} > 1, i = 1, 2 \tag{6.9}$$

The condition (6.9) is satisfied if and only if:

$$1 < \delta_i < \frac{\alpha + \beta + \alpha\beta - (1 + \beta)\gamma}{\alpha\beta} \text{ and } \gamma < \frac{\alpha + \beta}{1 + \beta} \tag{6.10}$$

for $0 < \alpha, \beta < 1$.

The combined system has a benefit factor of $f_i = \frac{(1+\alpha-\gamma)(1+\beta)}{1+\alpha\beta\delta_i}$ under the user behavior model B_i compared to the individual system. Figure 6.5 shows the benefit factor for some example α, β, and γ values, i.e., a lower bound on the expected effectiveness improvement by combining different reputation systems.

Thus, the benefit factor reaches the maximal value for $\delta_i \rightarrow 1, i = 1, 2$. This condition is equivalent to $s_{11} \approx s_{21}$ and $s_{12} \approx s_{22}$, or that the two systems have approximate resilience under any behavior model $\mathcal{B}_1$ and $\mathcal{B}_2$. In other words, *it is best to combine two systems with comparable performance against different behavior models to achieve the highest synergy among them.*

From Eq. (6.7), one can also derive a tighter bound between $\overline{s}_1$ and s_{11}, s_{21} than the one of Proposition 5. This new bound depends on the function $\Phi(\tau^*, \mathcal{T}^*, \tau'^*, \mathcal{T}'^*)$, which can be estimated numerically given certain assumptions on the shape of the error function $e(.)$ (Def. 1). In such a case, it it apparent that the resilience of the combined system (represented by $\overline{s}_1, \overline{s}_2$) is even better. The synergy achieved in this situation is dependent on the set of common users τ^* and related transactions $\mathcal{T}^*$. The thresholds of τ^* and τ'^* to ensure the combination is meaningful and effective, as clearly shown, are $\mid \tau^* \mid = \mid \tau'^* \mid > 0$.

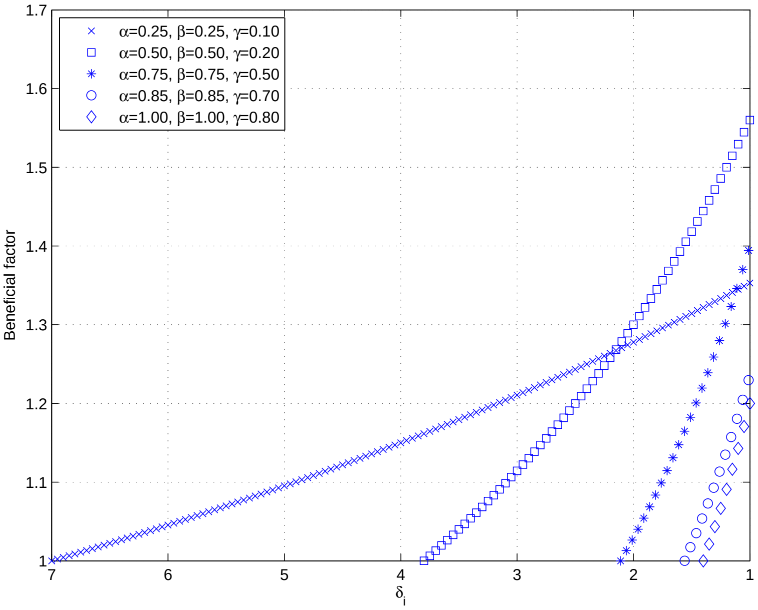


Figure 6.5: Benefit factor f_i vs. $\delta_i, i = 1, 2$

In summary, the total benefit (i.e, overall system resilience improvement under a mixture of behaviors) achieved by exchanging information and combining two computational trust models of two similar systems is contributed by two main factors: (1) The combined accuracy of two misbehavior detection algorithms, and (2) the effectiveness of the mechanisms for the reliable discovery of common users τ^* that enables effective reuse of user historical performance data for better evaluation of their behaviors.

The current analysis uses only a naive combination of two computation trust models from similar systems. There are also several effective methods of combining different computational trust models to guarantee better improvement in terms of evaluation user behaviors. Particularly promising solutions are the weighted majority algorithm (WMA) and boosting techniques (Freund & Schapire, 1995). The study of the usefulness of such techniques is beyond the scope of this chapter and subject to future work.

6.7 Conclusion

This chapter presents a first step towards the development of the next generation reputation-based trust management systems. Such systems are open in nature and are capable of acquiring and integrating knowledge from other similar systems to significantly improve its own performance in terms of detecting misbehavior and motivating higher level of cooperation among the users.

The chapter provides a systematically study on several opportunities of the development of these open systems. I have analyzed and quantified (based on a simple model) possible benefits of the combination of reputation data from two similar reputation systems and found that a synergy in performance of the two systems can be achieved even with a naïve combination of their trust models. This synergy comes from two main factors. The first source of performance improvement is from the combination of the trust evaluation mechanisms of the two systems, which yields an overall higher accuracy in detecting different types of misbehavior. The second comes from the discovery of common participants in the two systems, which contributes to the more reliable estimation of these users' trustworthiness. This work provides a starting point to the effective reuse and integration of trust-related data from many similar systems to improve the accuracy in the evaluation of user behaviors. I have also provided an extensive discussion of many challenges related to the development of an open approach to design a trust management system, which open up several interesting research questions for the trust research community.

Chapter 7

Conclusion and Future Work

7.1 What Has Been Done in this Thesis

Since the seminal work of (Marsh, 1994) that first attempts to formalize trust as a computational concept, trust management in open and decentralized systems such as peer-to-peer and social networks has become an active research area over the recent years. The research on trust and reputation in general has even a longer history, considering the enormous number of work in related disciplines, most notably sociology and economics research (Rousseau *et al.*, 1998).

Regarding the amount of existing works in these highly developed research areas over such a long time, the contributions of this thesis are in fact modest. These contributions should only be viewed and evaluated from the angle of developing and applying trust management techniques in online systems, and particularly in applications where participants are heterogenous computational agents with a pre-defined level of intelligence, e.g., with known goals and type of behaviors. Given the above research scope, this thesis provides and extends existing understanding of trust management in open peer-to-peer applications with respect to the following aspects.

First of all, the thesis provides an understanding of the different computational trust models from a machine-learning perspective (Chapter 3). Several (if not most) of existing computational approaches to trust evaluation can be seen as specially designed learning algorithms to estimate unknown parameters of different generative models of participant behaviors. Such generative models are usually built with expert domain knowledge and other ad-hoc heuristics. Under this new perspective, it is clarified that the design of computational trust models is in fact a subdiscipline of the development of machine learning techniques that are robust under different adversarial attacks. In other words, trust management research can be seen as AI research on security and robustness of machine learning techniques in presence of intelligent adversaries. Towards the comparison and evaluation of effectiveness of computational trust evaluation approaches, the thesis also discusses the prototyping and simulating of reputation-based trust systems in complex scenarios. An implementation of a general simulation framework for such purposes is also provided.

Secondly, my analysis on the relation of accuracy measures of a computational trust model and its capability to foster cooperation and establish trust among participants (Chapter 4) helps us to understand the key role of an identity management scheme in a heterogeneous environment where participants exhibit various types of behaviors and uses different trust management mechanisms to evaluate each other's reliability. The analysis in the case where cheap identities are possible reveals that pseudonyms, even though acquired at a low cost, actually have a high value to participants and thus appropriate pricing mechanisms can be implemented to enforce their cooperation in a variety of application scenarios where temporary gains by cheating are sufficiently bounded. This work also provides an approach to effectively use any existing computational trust mechanism with a given learning accuracy to promote cooperation in systems with mixed types of behaviors and possibilities of obtaining cheap identities.

Thirdly, I have shown that computational trust models can find applications in many different areas (Chapters 2 and 5). In the thesis two novel applications are introduced, namely the selection and ranking of services based on feedback of reputable users, and the use of specialized trust measures to choose reliable delegates in a key backup-recovery scenario in distributed online social networks.

Last but not least, the possibilities and challenges of building next-generation, open trust management systems are systematically studied (Chapter 6). This discussion and preliminary analysis has shown several possibilities of building such open systems, one of which is that information sharing among similar reputation-based trust systems may be beneficial in increasing the overall capability of individual systems sharing such information.

7.2 Open Issues and Future Directions

The work done in this thesis, like as any other research work, is by no means complete. Some of the open issues and future research directions are identified below, mostly as follow-up work of this thesis.

7.2.1 Towards an experimental approach to trust-related research

Trust management in more sophisticated systems, such as in those with real human participants, is of both practical and theoretical interest to the research community. These sophisticated systems may have an even higher level of heterogeneity and uncertain factors whose nature is not fully understood. More often participants are bounded rational, more indeterministic, and under the influence of numerous emotional and psychological factors. Given such complex settings, usually the problems become too complex and no long tractable mathematically. To discover and understand in-depth the various issues related to the cooperation and trust establishment in these practically sophisticated systems, an experimental approach is much more realistic.

Towards this research direction, there are various research issues related to the design, development and deployment of a large scale simulation platform that offers powerful and extensible capabilities of simulating various practical application scenarios with realistic assumptions. For example, the modeling of different attack models (Yang *et al.*, 2009) and their counter-measures should be sufficiently generic to simulate a wide range of behaviors and helps to discovery various vulnerabilities of the systems. This work can be based on existing work on attack modeling and simulation in network security, such as the Integrated Testbed project at CMU[1].

Furthermore, for enabling a realistic experimental approach to trust management, the emulation of practical and large scale usage scenarios is also highly desirable. Approaches to perform such large emulations automatically with reasonable deployment cost are needed. To this end, novel methods of using existing user communities on social networks and virtual environments such as Second Life[2] for emulation are a promising direction. These emulations, with participation of both real people and intelligent agents competing together, may provide better understanding of user behaviors and reveal more insights into the nature of cooperation and trust building in complex societies beyond the simple experiments usually conducted in current research work.

7.2.2 Towards open trust management systems

As discussed in Chapter 6, next-generation trust management systems are likely to follow an open architecture approach in terms of their capability to share, acquire, and integrate knowledge from other systems such as online social networks, recommender systems, search engines or similar trust systems.

Among the benefits of open systems, information sharing among similar trust management systems may help increase the overall capabilities of individual systems to detect malicious behaviors (Vu *et al.*, 2009b). Also, under realistic assumptions, this sharing may help to increase the total adversarial cost to attack the systems and thus strengthening significantly (Vu *et al.*, 2010).

There are several research issues related to the building of such open systems, as discussed in detail in Chapter 6. A number of these open questions are readily answered with known approaches and techniques while having the potentials to provide good results with practical impacts. First, a promising direction is to study the combination of different computational models from different systems to improve their capabilities of detecting malicious behavior, using conventional machine learning techniques such as the Weight Majority Algorithms and boosting (Freund & Schapire, 1995).

Second, it is possible to extend existing entity resolution techniques, e.g., entity matching and detection of user's digital footprints via Friend-of-Friend networks (Shen *et al.*, 2005b), to build a Web identity management that is robust against cold start and whitewashing behaviors. The reason is a Web identity management may exploit information from many information sources and thus it becomes much more difficult for users to manipulate their activity trails.

Third, it is also interesting to analyze users' incentives of cooperation, considering the privacy concerns, where similar systems exchange information among each other. Privacy-preserving data collection and mining techniques (Clifton *et al.*, 2004; Lindell & Pinkas, 2002) may be applicable for such purposes. Certain cryptogrphic techniques such as homomorphic

[1] http://www.truststc.org/testbeds.htm

[2] secondlife.com

encryption (Prabhakaran & Rosulek, 2008) can also be used, for example as in (Domingo-Ferrer *et al.*, 2008). However, regarding their high computational cost, lighter weight approaches such as anonymized techniques based on obfuscation may be preferable.

7.2.3 Trust management in the digital economy and beyond

The human society has become more and more sophisticated with the increasing merge between the virtual realms and the physical realities. Business transactions are increasingly carried out on the Web across the boundary of virtual society and real environments, e.g., considering the economy in massively multi-player online games such as Second Life or World of Warcraft. Moreover, the next generation of the Internet itself may evolve into a social system that relies on social relationships among people on the application layer rather than on the physical network level, e.g., the Davis Social Links project (Banks *et al.*, 2007). This vision may become closer to reality, given that with the current technological advances, the binding between users and devices is much tighter. The management of online identities (or avatars somedays) and real-life identities in doing business-related transactions will become more comfortable, yet with potentially higher risks. The notion of a digital economy will become ubiquitous and more social than ever.

Trust mechanisms have been used in various Internet business applications and virtual communities such as online social networks, e.g., (Caverlee *et al.*, 2010), to determine the reliability of prospective business partners. In an ubiquitous digital economy research on trust management would be more than a necessity to provide in-depth understanding and to predict various phenomena regarding the cooperation among participants. Trust management in these new application contexts will pose much more significant challenges as the systems under study have an even higher level of heterogeneity, and the environments exhibit extremely high uncertainty whose nature may yet be fully understood.

Among the questions that need to be addressed when applying trust-related research in the digital economy, the most relevant one is possibly the study of the incentive structure of participating users. User incentives can be analyzed via emulation of the systems in realistic conditions (see Section 7.2.1), or by other formal methods such as eliciting user preferences via intelligent user interfaces. Understanding and giving the right incentives to users plays a key role in ensuring their cooperation. It also helps to build a secure and robust system against possible malicious behaviors, as users simply reject security guidelines according to their own rational analysis (unknown to the system designer) (Herley, 2009).

Privacy is another important issue that has not been considered in this thesis. As information on activities of participants should be shared in order to enable any users to estimate reliability and trustworthiness of another, certain privacy concerns emerge. There are few works (Pavlov *et al.*, 2004; Xiong, 2005) that attempt to answer some aspects of the question, particularly on how to aggregate and compute reputation in a privacy-preserving way, e.g., using homomorphic encryption. It is still not understood how much reputation information should be shared among peers so as to achieve highest cooperation with the least information leakage. The question is even more challenging if users have different levels of privacy awareness and exhibiting different types of behaviors, including rational and strategically malicious.

Even though the study of trust and cooperation in a general human society is highly complex, we can rely on a large amount of work from other related disciplines regarding these issues. For example, sociologists and economists have spent significant amount of effort studying various issues related to reputation and cooperation in different types of economies (Rousseau *et al.*, 1998). In a variety of online applications things may be simpler as autonomous intelligent agents acting on behalf of users, e.g., the vision of the Semantic Web (Berners-Lee *et al.*, 2001), are eventually available, and as humans may behave more rationally (optimally) in a near future. Given that computational means for humans are becoming more available, decision-making tools are ubiquitous and people are educated to think and act more rationally when finding solutions to their everyday tasks (the computational thinking paradigm (Wing, 2006)). Therefore, we may expect that several existing analysis and mechanisms from micro-economics and operations research, e.g., the idea of using a market of reputation (Tadelis, 1999, 2002), may be reusable in these new contexts.

Appendix A

Appendix: Example of Probabilistic Inference Using the Junction Tree Algorithm

In this appendix we briefly describe the Junction Tree Algorithm (JTA) to do the probabilistic inference on a directed acyclic graphical model to compute the distribution of certain variables. For this example, we use the example model of the dependencies among the different QoS parameters and on the different contextual factors as in Figure A.1 (a).

Suppose that the peer P_0 (on behalf of a user) would like to know the probability that the peer P providing the file hosting service with the download speed level $D = high$, given that the user only pays for the most economical type of service, $P = economic$. This is equivalent to the computation of the conditional probability: *p(D=high |P=economic)*, or briefly written as $p(D = high|P^*)$. The notation P^* denote the claiming of the variable P to its associated evidential state $P = economic$. The computation of such probability is done via the JTA as follows. Firstly, we create the Junction Tree from the original model in Figure A.1 (a) via the four steps:

- *Moralization of the dependency graph*: add a link between any two parents with the same child node and having no direct link between them; afterwards, make the graph to be undirected. This is done as in Figure A.1 (a).
- *Triangulation of the moralized graph*: any loop of length 4 or more must have a link between its two non-consecutive vertices. The moralized graph in Figure A.1 (b) has been triangulated by itself.
- *Building the Clique Tree*: identify any clique (maximal complete sub-graph) of the triangulated tree and the intersection between any two cliques (called a separator). This will form a clique tree, as in Figure A.1 (c).
- *Building the Junction Tree*: Remove any redundant edges with the same separator in any loop such that in the resulted tree, for each pair of vertices x and y , all vertices on the path between them contain the intersection $x \cap y$ (the Running Intersection Property).

This step is illustrated in Figure A.1 (d).

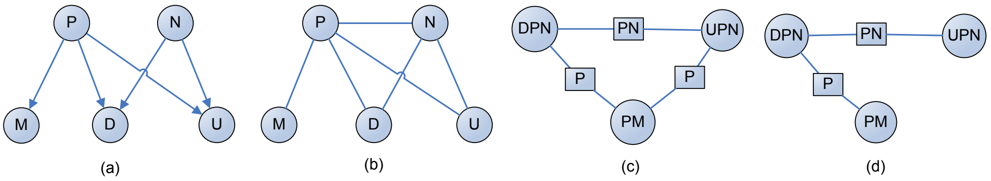


Figure A.1: The example graphical model for QoS parameters of the file hosting service (a) before and (b) after the moralization step and the triangulation step. (c) The building of the clique tree; (d) We remove the redundant edge *PM-P-UPN* to get the junction tree of the model.

Secondly, the joint distribution of the original network of Figure A.1 (a) could be written as in Equation (A.1), where X^* stands for the set of all variables P^*, N, U, D, M:

$$p(X^*) = p(P^*)p(N)p(D|P^*N)p(U|P^*N)p(M|P^*) \tag{A.1}$$

On the other hand, we could also write this distribution in terms of the potentials $\phi(.)$ of the cliques and separators of the junction tree in Figure A.1 (d):

$$p(X^*) = \frac{\phi(DP^*N)\phi(UP^*N)\phi(P^*M)}{\phi(P^*)\phi(P^*N)} \tag{A.2}$$

One way to assign the initial potentials for the cliques and separators in (A.2) such that they also satisfy (A.1) is: $\phi(DP^*N) = p(P^*)p(N)P(D|P^*N); \phi(UP^*N) = p(U|P^*N); \phi(P^*M) = p(M|P^*)$; and $\phi(M^*) = \phi(P^*N) = 1.0$.

We then perform the message passing on the built junction tree as illustrated in Figure A.2. During the passing of the messages, the potential functions of the cliques and separators will be modified as follows: message 1 corresponds to the absorbtion of the clique DP^*N from the clique UP^*N via the separator P^*N, leading to the following potential updates:

$$\phi^*(P^*N) \quad = \quad \sum_{UP^*N \setminus P^*N} \phi(UP^*N) = \sum_{U} p(U|P^*N) = 1.0$$

$$\phi^*(DP^*N) \quad = \quad \phi(DP^*N)\frac{\phi^*(P^*N)}{\phi(P^*N)} = p(P^*)p(N)P(D|P^*N)\frac{1.0}{1.0} = p(P^*)p(N)P(D|P^*N)$$

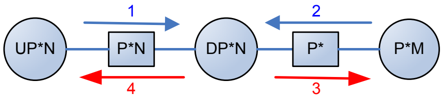


Figure A.2: The running of the JTA by the passing of messages in the specified order.

The process is performed similarly with the other messages $2, 3$ and 4 (in this order). For example, message 2 corresponds to the updates:

$$\phi^*(P^*) = \sum_{P^*M\setminus P^*} \phi(P^*M) = \sum_M p(M|P^*) = 1.0$$

$$\phi^{**}(DP^*N) = \phi^*(DP^*N)\frac{\phi^*(P^*)}{\phi(P^*)} = p(P^*)p(N)P(D|P^*N)\frac{1.0}{1.0} = p(P^*)p(N)P(D|P^*N) \tag{A.3}$$

At the end of the algorithm, the value of the potential function of each clique and separator would be the marginal probability of that very clique (or separator). Thus the probability that a user has a high download speed, given that it only pays an economical price is $p(D = high|P^*) \propto \sum_N p(D = high, P^*N) = \sum_N \phi^{**}(D = high, P^*N)$, where the term $\phi^{**}(D = high, P^*N)$ has already been computed after the passing of message 2 in Equation (A.3). One important aspect is that the whole above procedure can be done automatically with any pre-defined dependency graph, given the conditional probability entries of all nodes, of which the learning can be done automatically as well. This enables the use of our quality and trust computational framework in much wider application scenarios.

References

(2004). *DIP Integrated project- Data, Information, and Process Integration with Semantic Web Services*. http://dip.semanticWeb.org/. 110

(2008). *Amazon Web Services @ Amazon.com*. http://aws.amazon.com, last accessed October 2008. 111

(2008). *Workflow Management Coalition (WfMC)*. http://www.wfmc.org/, last accessed October 2008. 111

A. Baker et al (2007). Reputation-based systems: a security analysis. Tech. rep., European Network on Information Security Agency (ENISA). 3, 18

Abdul-Rahman, A. (2005). *A Framework for Decentralised Trust Reasoning*. Ph.D. thesis, University College London, Department of Computer Science. 4, 6

Aberer, K. & Despotovic, Z. (2001). Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, 310–317, ACM Press, New York, NY, USA. 16, 24, 48, 58, 60, 73, 75, 93, 114

Aberer, K. & Hauswirth, M. (2003). Start making sense: The chatty web approach for global semantic agreements. *Journal of Web Semantics*, **1**, 2003. 151

Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M. & Schmidt, R. (2003a). P-Grid: a self-organizing structured P2P system. *SIGMOD Rec.*, **32**, 29–33. 69

Aberer, K., Cudré-Mauroux, P. & Hauswirth, M. (2003b). The chatty web: Emergent semantics through gossiping. In *Proceedings of WWW*. 151

Aberer, K., Cudré-Mauroux, P., Ouksel, A.M., Catarci, T., Hacid, M.S., Illarramendi, A., Kashyap, V., Mecella, M., Mena, E. & Neuhold, E.J. (2004). Emergent semantics principles and issues. In *Database Systems for Advances Applications (DASFAA 2004), Proceedings*, 25–38, Springer. 151

Adler, B.T. & de Alfaro, L. (2007). A content-driven reputation system for the wikipedia. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 261–270, ACM, New York, NY, USA. 18

Agrawal, R. & Terzi, E. (2006). On honesty in sovereign information sharing. In *Proc. of EDBT'06*, vol. 3896, 240–256, Springer. 107

Alon, N. (1998). Spectral techniques in graph algorithms. In *LATIN '98: Proceedings of the Third Latin American Symposium on Theoretical Informatics*, 206–215, Springer-Verlag, London, UK. 134

Anceaume, E. & Ravoaja, A. (2006). Incentive-based robust reputation mechanism for p2p services. In *OPODIS'06*, 305–319. 65

Ashri, R., Ramchurn, S.D., Sabater, J., Luck, M. & Jennings, N.R. (2005). Trust evaluation through relationship analysis. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 1005–1011. 17, 58, 71

Avenhaus, R., Stengel, B.V. & Zamir, S. (2002). Inspection games. *Hand-*

book of Game Theory with Economic Applications, **3**, 1947–1987, available at http://ideas.repec.org/h/eee/gamchp/3-51.html. 67, 107

Avesani, P., Massa, P. & Tiella, R. (2005). A trust-enhanced recommender system application: Moleskiing. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, 1589–1593, ACM Press, New York, NY, USA. 18

Axelrod, R. & Hamilton, W. (1981). The evolution of cooperation. *Science*, **211**, 1390–1396. 2

Backstrom, L., Dwork, C. & Kleinberg, J. (2007). Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 181–190, ACM, New York, NY, USA. 156, 160

Banks, L., Ye, S., Huang, Y. & Wu, S. (2007). Davis social links: Integrating social networks with internet routing. In *Proceedings of the 2007 workshop on Large scale attack defense*, 121–128, ACM. 169

Berners-Lee, T., Hendler, J. & Lassilar, O. (2001). The Semantic Web. *Scientific American*, **284**, 34–43. 112, 170

Bhattacharjee, R. & Goel, A. (2005). Avoiding ballot stuffing in ebay-like reputation systems. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, 133–137, ACM, New York, NY, USA. 99

Bhatti, R., Bertino, E. & Ghafoor, A. (2007). An integrated approach to federated identity and privilege management in open systems. *Commun. ACM*, **50**, 81–87. 150

Bilgin, A.S. & Singh, M.P. (2004). A DAML-based repository for QoS-aware semantic web service selection. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, 368–375, IEEE Computer Society, Washington, DC, USA. 113

Blaze, M., Feigenbaum, J. & Lacy, J. (1996). Decentralized trust management. *Security and Privacy, IEEE Symposium on*, **0**, 0164. 12, 14

Bryce, C., Dimmock, N., Krukow, K., Seigneur, J.M., Cahill, V. & Wagealla, W. (2005). Towards an evaluation methodology for computational trust systems. In *Proceedings of Third International Conference on Trust Management, Rocquencourt, France*, 289–304. 17

Buchegger, S. & Boudec, J.Y.L. (2004). A robust reputation system for P2P and mobile ad-hoc networks. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*. 18, 24, 25, 48

Buchegger, S. & Datta, A. (2009). A case for P2P infrastructure for social networks - opportunities and challenges. In *Proceedings of WONS 2009, The Sixth International Conference on Wireless On-demand Network Systems and Services*, Snowbird, Utah, USA. 128

Buchegger, S., Schiöberg, D., Vu, L.H. & Datta, A. (2009). PeerSoN: P2P social networking - early experiences and insights. In *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*, Nürnberg, Germany. 128, 132

Bui, T. & Jones, C. (1992). Finding good approximate vertex and edge partitions is np-hard. *Inf. Process. Lett.*, **42**, 153–159. 134

Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, **8**, 195–210. 39, 43

Buyya, R., Stockinger, H., Giddy, J. & Abramson, D. (2001). Economic models for management of resources in peer-to-peer and grid computing. In *Proceedings of the SPIE International Conference on Commercial Applications for High-Performance Computing*, Denver, USA. 70

Carminati, B. & Ferrari, E. (2008). Privacy-aware collaborative access control in web-based social networks. In *DBSec*, 81–96. 145

Caverlee, J., Liu, L. & Webb, S. (2010). The socialtrust framework for trusted social information management: Architecture and algorithms. *Information Sciences*, **180**, 95 – 112, special Issue on Collective Intelligence. 169

Chakraborty, S. & Ray, I. (2006). Trustbac: integrating trust relationships into the rbac model for access control in open systems. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, 49–58, ACM, New York, NY, USA. 145

Chapin, P.C., Skalka, C. & Wang, X.S. (2008). Authorization in trust management: Features and foundations. *ACM Comput. Surv.*, **40**, 1–48. 13, 14, 15

Chen, Z., Liang-Tien, C., Silverajan, B. & Bu-Sung, L. (2003). UX - an architecture providing QoS-aware and federated support for UDDI. In *Proceedings of the IEEE International Conference on Web Services (ICWS'03)*. 113, 114

Clifton, C., Kantarcioğlu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A. & Suciu, D. (2004). Privacy-preserving data integration and sharing. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 19–26, ACM, New York, NY, USA. 168

Cornelli, F., Damiani, E., Vimercati, S.C., Paraboschi, S. & Samarati, P. (2002). Choosing reputable servents in a P2P network. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, 376–386, ACM Press, New York, NY, USA. 71, 114

Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K. & de Meer, H. (2009). idMesh: Graph-Based Disambiguation of Linked Data Engines. In *WWW'09: Proceedings of the 18th International World Wide Web conference*, Madrid, Spain. 160

Cutillo, L.A., Molva, R. & Strufe, T. (2009). Privacy preserving social networking through decentralization. In *WONS 2009, 6th International Conference on Wireless On-demand Network Systems and Services, February 2-4, 2009, Snowbird, Utah, USA*. 128

Dasgupta, S. (1997). The sample complexity of learning fixed-structure bayesian networks. *Mach. Learn.*, **29**, 165–180. 43

Datta, A., Hauswirth, M. & Aberer, K. (2003). Beyond "web of trust": enabling p2p e-commerce. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, 303–312. 69

Day, J. & Deters, R. (2004). Selecting the best web service. In *Proceedings of the IBM Centers for Advanced Study Conference (CASCON '04)*, 293–308. 114

Dellarocas, C. (2003). Efficiency and robustness of binary feedback mechanisms in trading environments with moral hazard. Available at http://ideas.repec.org/p/mit/sloanp/1852.html. 60

Dellarocas, C. (2004). *Strategic Manipulation of Internet Opinion Forums: Implications for Consumers and Firms*. Working Paper. 16

Dellarocas, C. (2005a). Reputation mechanism design in online trading environments with pure moral hazard. *Information Systems Research*, **16**, 209–230. 16, 48, 65, 68, 83, 107, 108

Dellarocas, C. (2005b). Reputation Mechanisms. *Handbook on Economics and Information Systems (T. Hendershott, ed.), Elsevier Publishing*. 3, 15, 16, 24, 72, 107

Despotovic, Z. (2005). *Building trust-aware P2P systems*. Ph.D. thesis, Swiss Federal Institute of Technology Lausanne, Switzerland. 6, 12, 65, 70

Despotovic, Z. & Aberer, K. (2004a). Possibilities for managing trust in P2P networks. Tech. Rep. IC200484, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland. 112, 113, 116

Despotovic, Z. & Aberer, K. (2004b). A probabilistic approach to predict peers' performance in P2P networks. In M. Klusch, S. Ossowski, V. Kashyap & R. Unland, eds., *Cooperative Information Agents VIII: 8th International Workshop, CIA 2004*, vol. 3191 / 2004, 62–76. 24, 25, 48, 93

Despotovic, Z. & Aberer, K. (2005). Probabilistic prediction of peersŠ performance in P2P networks. *Engineering Applications of Artificial Intelligence*, **18**, 771–780. 12, 60

Despotovic, Z. & Aberer, K. (2006). P2P reputation management: Probabilistic estimation vs. social networks. *Journal of Computer Networks, Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security*, **50**, 485–500. 11, 15, 16, 24, 48, 72, 107, 156

Ding, C., Yueguo, C. & Weiwei, C. (2009). A survey study on trust management in p2p systems. 15, 16

Dobson, G. (2004). *Quality of Service in Service-Oriented Architectures*. http://digs.sourceforge.net/papers/qos.html. 113

Domingo-Ferrer, J., Viejo, A., Sebe, F. & Gonzalez-Nicolas, U. (2008). Privacy homomorphisms for social networks with private relationships. *Computer Networks (in press, available online 11 July 2008)*. 145, 169

Douceur, J.R. (2002). The sybil attack. In *IPTPS'02*, 251–260. 3, 18, 66, 67, 149

Emekci, F., Sahin, O.D., Agrawal, D. & Abbadi, A.E. (2004). A peer-to-peer framework for web service discovery with ranking. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, 192, IEEE Computer Society, Washington, DC, USA. 114

Foster, I. & Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 111

Fran c.L., Mé, L. & Tong, V.V.T. (2008). A distributed certification system for structured P2P networks. In *Proceedings of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, vol. 5127 of *Lecture Notes in Computer Science*, 40–52, Springer-Verlag, Bremen, Germany. 144, 145

Freund, Y. & Schapire, R.E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, 23–37, Springer-Verlag, London, UK. 158, 165, 168

Friedman, E.J. & Resnick, P. (2001). The social cost of cheap pseudonyms. *Journal of Economics & Management Strategy*, **10**, 173–199. 3, 66, 67, 87, 107

Fullam, K., Klos, T., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K.S., Rosenschein, J., Vercouter, L. & Voss, M. (2005). A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 512–518. 7, 17, 50, 52

Galuba, W., Aberer, K., Despotovic, Z. & Kellerer, W. (2007). Authentication-free fault-tolerant peer-to-peer service provisioning. In *Databases, Information Systems and Peer-to-Peer Computing, DBISP2P 2007*, Springer. 18

Galuba, W., Aberer, K., Despotovic, Z. & Kellerer, W. (2009). ProtoPeer: A P2P Toolkit Bridging the Gap Between Simulation and Live Deployment. In *2nd International Conference on Simulation Tools and Techniques (SIMUTools'09)*. 51

Golbeck, J. (2006). Trust on the world wide web: A survey. *Foundations and Trends in Web Science*, **1**, 131–197. 11, 15, 16, 72, 107, 148

Golbeck, J. & Hendler, J.A. (2004). Reputation network analysis for email filtering. In *CEAS*. 18

Graham, P. (2002). *A plan for spam, Hackers and Painters, Big Ideas from the Computer Age*. OŠReally, 2004. 71, 73

Guha, R., Kumar, R., Raghavan, P. & Tomkins, A. (2004). Propagation of trust and distrust. In *International World Wide Web Conference (WWW2004)*. 48, 114

Gummadi, K.P., Saroiu, S. & Gribble, S.D. (2002). King: estimating latency between arbitrary internet end hosts. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 5–18, ACM Press, New York, NY, USA. 55, 90

Gupta, R. & Somani, A.K. (2004). Reputation management framework and its use as currency in large-scale peer-to-peer networks. *p2p*, **00**, 124–132. 17

Gutscher, A. (2007). A trust model for an open, decentralized reputation system. In *Proceedings of the Joint iTrust and PST Conferences on Privacy Trust Management and Security (IFIPTM 2007)*. 150

Gyongyi, Z., Garcia-Molina, H. & Pedersen, J. (2004). Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, 271–279. 18

Herley, C. (2009). So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the New Security Paradigms Workshop 2009*, ACM. 169

Hsu, J.Y.J., Lin, K.J., Chang, T.H., Ho, C.J., Huang, H.S. & Jih, W.R. (2006a). Parameter learning of personalized trust models in broker-based distributed trust management. *Information Systems Frontiers*, **8**, 321–333. 17, 48

Hsu, J.Y.J., Lin, K.J., Chang, T.H., Ho, C.J., Huang, H.S. & Jih, W.R. (2006b). Parameter learning of personalized trust models in broker-based distributed trust management. *Information Systems Frontiers*, **8**, 321–333. 24

Huang, C. & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, **15**, 225–263. 29, 40, 41

Jamali, M. & Ester, M. (2009). Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 397–406, ACM, New York, NY, USA. 18

Jim, T. (2001). Sd3: A trust management system with certified evaluation. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 106, IEEE Computer Society, Washington, DC, USA. 15

Jøsang, A., Hayward, R. & Pope, S. (2006). Trust network analysis with subjective logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, 85–94, Australian Computer Society, Inc., Darlinghurst, Australia, Australia. 12

Jøsang, A., Ismail, R. & Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, **43**, 618–644. 11, 15, 16, 24, 48, 72, 107, 112, 113, 148

Jurca, R. (2007). *Truthful reputation mechanisms for online systems*. Ph.D. thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. 2, 6, 16

Jurca, R. & Faltings, B. (2006). Minimum payments that reward honest reputation feedback. In *Proceedings of the ACM Conference on Electronic Commerce*, 190–199, Ann Arbor, Michigan, USA. 65

Kalepu, S., Krishnaswamy, S. & Loke, S.W. (2004). Reputation = f(user ranking, compliance, verity). In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, 200, IEEE Computer Society, Washington, DC, USA. 18, 113, 114

Kamvar, S.D., Schlosser, M.T. & Molina, H.G. (2003). The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of WWW'03*. 12, 16, 24, 48, 58, 65, 73, 75, 78, 93, 107

Kinateder, M., Baschny, E. & Rothermel, K. (2005). Towards a generic trust model - comparison of various trust update algorithms. In *Proceedings of Third International Conference on Trust Management, Rocquencourt, France*, 177–192. 17

Kiyavash, N. & Koushanfar, F. (2007). Anti-Collusion Position Estimation in Wireless Sensor Networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 1–9, Pisa, Italy. 19

Klemm, A., Lindemann, C., Vernon, M.K. & Waldhorst, O.P. (2004). Characterizing the query behavior in peer-to-peer file sharing systems. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 55–67, ACM, New York, NY, USA. 59

Kohlas, R. (2007). *Decentralized trust evaluation and public-key authentication*. Ph.D. thesis, University of Bern, Switzerland. 6

KRUKOW, K. (2006). *Towards a theory of trust for the global ubiquitous computer*. Ph.D. thesis, Department of Computer Science, University of Aarhus, Denmark. 6

LANDA, R., GRIFFIN, D., CLEGG, R.G., MYKONIATI, E. & RIO, M. (2009). A sybilproof indirect reciprocity mechanism for peer-to-peer networks. In *INFOCOM 2009. The 28th Conference on Computer Communications. IEEE*, 343–351. 18

LEVIEN, R. (2002). *Attack-resistant trust metrics*. Ph.D. thesis, University of California at Berkeley. 6, 78, 107

LIANG, Z. & SHI, W. (2005). PET: A PErsonalized Trust Model with Reputation and Risk Evaluation for P2P Resource Sharing. In *HICSS*. 19

LIANG, Z. & SHI, W. (2008). Analysis of ratings on trust inference in open environments. *Perform. Eval.*, **65**, 99–128. 17, 52, 65, 66

LINDELL, Y. & PINKAS, B. (2002). Privacy preserving data mining. *Journal of cryptology*, **15**, 177–206. 168

LIU, Y., NGU, A. & ZHENG, L. (2004). QoS computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 66–73, ACM Press, New York, NY, USA. 34, 114

MAN AU YEUNG, C., LICCARDI, I., LU, K., SENEVIRATNE, O. & BERNERS-LEE, T. (2009). Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*. 128

MARSH, S.P. (1994). *Formalising Trust as a Computational Concept*. Ph.D. thesis, Department of Mathematics and Computer Science, University of Stirling. 1, 5, 166

MASSA, P. & AVESANI, P. (2004). Trust-aware collaborative filtering for recommender systems. In *CoopIS/DOA/ODBASE (1)*, 492–508. 18

MAXIMILIEN, E.M. & SINGH, M.P. (2002). Reputation and endorsement for web services. *SIGecom Exch.*, **3**, 24–31. 18, 112, 114

MAXIMILIEN, E.M. & SINGH, M.P. (2005). Agent-based trust model involving multiple qualities. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 519–526, ACM Press, New York, NY, USA. 18, 34

MILLER, N., RESNICK, P. & ZECKHAUSER, R. (2005). Eliciting informative feedback: The peer-prediction method. *Management Science*, **51**, 1359–1373. 16, 65, 75, 85

MUI, L. (2002). *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. Ph.D. thesis, MIT. 2, 6

MUI, L., MOHTASHEMI, M. & HALBERSTADT, A. (2002). A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Science (HICSS)*, Hawaii, USA. 24, 25, 48

MURPHY, K. (2007). *Bayes Net Toolbox for Matlab*. http://bnt.sourceforge.net/, last access October 2008. 43

NAICKEN, S., BASU, A., LIVINGSTON, B. & RODHETBHAI, S. (2006). A survey of peer-to-peer network simulators. *Proceedings of the 7th Annual Postgraduate Symposium (PGNet '06)*. 51

NARAYANAN, A. & SHMATIKOV, V. (2009). De-anonymizing social networks. *IEEE Security* *and Privacy (to appear)*. 156, 160

Neal, R. & Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, ed., *Learning in Graphical Models*, Kluwer. 28, 29, 39

North, M.J., Collier, N.T. & Vos, J.R. (2006). Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, **16**, 1–25. 55, 59, 138

O'Sullivan, J., Edmond, D. & ter Hofstede, A.H.M. (2005). Formal description of non-functional service properties. Tech. rep., Business Process Management Group, Centre for Information Technology Innovation, Queensland University of Technology, Australia. 111

Ouzzani, M. & Bouguettaya, A. (2004). Efficient access to Web services. *IEEE Internet Computing*, 34–44. 112, 114, 120

Palomar, E., Tapiador, J.M.E., Hernandez-Castro, J.C. & Ribagorda, A. (2008). Secure content access and replication in pure p2p networks. *Comput. Commun.*, **31**, 266–279. 145

Paolucci, M., Kawamura, T., Payne, T.R. & Sycara, K.P. (2002). Semantic matching of web services capabilities. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, 333–347, Springer-Verlag, London, UK. 120

Papazoglou, M.P. & Georgakopoulos, D. (2003). Service-oriented computing. *Commun. ACM*, **46**, 24–28. 70

Patel, C., Supekar, K. & Lee, Y. (2003). A QoS oriented framework for adaptive management of web service based workflows. In *Proceeding of Database and Expert Systems 2003 Conference*, 826–835. 112, 114

Patel, J., Teacy, W.T.L., Jennings, N.R. & Luck, M. (2005). A probabilistic trust model for handling inaccurate reputation sources. In P. Herrmann, V. Issarny & S. Shiu, eds., *Proceedings of iTrust'05, France*, 193–209, Springer Berlin Heidelberg. 24, 25, 48, 65, 66

Pavlov, E., Rosenschein, J.S. & Topol, Z. (2004). Supporting privacy in decentralized additive reputation systems. In *iTrust*, 108–119. 169

Pingel, F. & Steinbrecher, S. (2008). Multilateral secure cross-community reputation systems for internet communities. In *TrustBus '08: Proceedings of the 5th international conference on Trust, Privacy and Security in Digital Business*, 69–78, Springer-Verlag, Berlin, Heidelberg. 150

Prabhakaran, M. & Rosulek, M. (2008). Towards robust computation on encrypted data. In *ASIACRYPT*, vol. 5350 of *Lecture Notes in Computer Science*, 216–233, Springer. 169

Ran, S. (2003). A model for Web services discovery with QoS. *SIGecom Exch.*, **4**, 1–10. 112, 114, 116

Regan, K., Cohen, R. & Poupart, P. (2006). Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. 17, 24, 48

Rehák, M. & Pechoucek, M. (2007). Trust modeling with context representation and generalized identities. In *CIA*, 298–312. 150

Resnick, P., Kuwabara, K., Zeckhauser, R. & Friedman, E. (2000). Reputation systems. *Commun. ACM*, **43**, 45–48. 23, 48, 69

Rettinger, A., Nickles, M. & Tresp, V. (2008). A statistical relational model for trust learning. In *AAMAS (2)*, 763–770. 17

Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I. & Yu, H. (2005). OpenDHT: a public DHT service and its uses. In *SIGCOMM '05*. 132

Richardson, M., Agrawal, R. & Domingos, P. (2003). Trust management for the Semantic Web. *Lecture Notes in Computer Science*, 351–368. 114

Roddick, J.F., Hornsby, K. & de Vries, D. (2003). A unifying semantic distance model for determining the similarity of attribute values. In *ACSC '03: Proceedings of the 26th Australasian computer science conference*, 111–118, Australian Computer Society, Inc., Darlinghurst, Australia, Australia. 157

Rousseau, D., Sitkin, S., Burt, R. & Camerer, C. (1998). Not so different after all: A cross-discipline view of trust. *Academy of management review*, **23**, 393–404. 166, 170

Sabater, J. (2003a). *Trust and reputation for agent societies*. Ph.D. thesis, Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Catalonia. 6

Sabater, J. (2003b). *Trust and Reputation for Agent Societies*. No. 20 in Monografies de l'institut d'investigació en intel.ligència artificial, IIIA-CSIC. 52

Salem, N.S., Hubaux, J.P. & Jakobsson, M. (2005). Reputation-based wi-fi deployment. *SIGMOBILE Mob. Comput. Commun. Rev.*, **9**, 69–81. 19

Sandhu, R. & Zhang, X. (2005). Peer-to-peer access control architecture using trusted computing technology. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, 147–158, ACM, New York, NY, USA. 145

Saxena, N., Tsudik, G. & Yi, J.H. (2007). Threshold cryptography in p2p and manets: The case of access control. *Comput. Netw.*, **51**, 3632–3649. 144

Schlosser, A., Voss, M. & Brückner, L. (2005). On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation*, **10**. 17, 52, 57, 65

Schneier, B. (1995). *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA. 128, 130, 144

Schosser, S., Böhm, K. & Vogt, B. (2008). Do humans identify efficient strategies in structured peer-to-peer systems? In *AAMAS (3)*, 1517–1520. 17

Shamir, A. (1979). How to share a secret. *Commun. ACM*, **22**, 612–613. 128

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B. & Mei, H. (2007). Personalized qos prediction forweb services via collaborative filtering. *icws*, **0**, 439–446. 34

Shen, W., Li, X. & Doan, A. (2005a). Constraint-based entity matching. In *Proceedings of the AAAI*. 151, 156

Shen, W., Li, X. & Doan, A. (2005b). Constraint-based entity matching. In *Proceedings of the AAAI*. 168

Shubha Kher, R.G. & Somani, A.K. (2005). Network selection using fuzzy logic. In *2nd International Conference on Broadband Networks*, 941–950. 19

Srinivasan, N., Paolucci, M. & Sycara, K.P. (2004). Adding OWL-S to UDDI, implementation and throughput. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, USA. 121

Srivatsa, M., Xiong, L. & Liu, L. (2005). Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, 422–431, ACM Press, New York, NY, USA. 107

Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, 149–160, ACM Press, New York, NY, USA. 43, 139

Stutzbach, D. & Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, 189–202, ACM Press, New York, NY, USA. 55, 59, 90, 91

Sun, Y.L., Han, Z., Yu, W. & Liu, K.J.R. (2006). A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In *INFOCOM'06*. 65

Suryanarayana, G., Diallo, M.H., Erenkrantz, J.R. & Taylor, R.N. (2006). Architectural support for trust models in decentralized applications. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, 52–61, ACM Press, New York, NY, USA. 17, 51

Tadelis, S. (1999). What's in a name? reputation as a tradeable asset. *American Economic Review*, **89**, 548–563. 154, 170

Tadelis, S. (2002). The market for reputations as an incentive mechanism. *Journal of Political Economy*, **110**, 854–882. 170

Teacy, W. (2006). *Agent-based trust and reputation in the context of inaccurate information sources*. Ph.D. thesis, Citeseer. 6

Tian, M., Gramm, A., Naumowicz, T., Ritter, H. & Schiller, J. (2003). A concept for QoS integration in web services. In *Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops*, vol. 00, 149–155, Italy. 112, 114, 116

Twigg, A. (2003). A subjective approach to routing in P2P and ad-hoc networks. In *Proceeding of iTrust*, 225–238. 18

Varia, J. (2008). *Cloud Architectures white paper*. http://jineshvaria.s3.amazonaws.com/public/cloudarchitec last accessed October 2008. 111

von der Weth, C. & Böhm, K. (2007). Towards an objective assessment of centrality measures in reputation systems. In *CEC/EEE*, 270–277. 18

von Laszewski, G., Alunkal, B. & Veljkovic, I. (2005). Toward Reputable Grids. *Scalable Computing: Practice and Experience*, **6**, 95–106. 18

Vu, L.H. & Aberer, K. (2006). Probabilistic estimation of peers' quality and behaviors for subjective trust evaluation. Tech. Rep. LSIR-REPORT-2006-013. 23

Vu, L.H. & Aberer, K. (2007). A probabilistic framework for decentralized management of trust and quality. In *CIA*, 328–342. 17, 23, 60, 72

Vu, L.H. & Aberer, K. (2008a). Effective usage of computational trust models in rational environments. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, **1**, 583–586. 64, 79

Vu, L.H. & Aberer, K. (2008b). Effective usage of computational trust models in rational environments. Tech. Rep. LSIR-REPORT-2008-007, available at http://infoscience.epfl.ch/search?recid=125277&of=hd. 64, 157

Vu, L.H. & Aberer, K. (2009). Towards Probabilistic Estimation of Quality of Online Services. In *2009 IEEE International Conference on Web Services (ICWS 2009)*, IEEE. 17, 23

Vu, L.H. & Aberer, K. (200x). Effective usage of computational trust models in rational environments. *in third round review of the ACM Transaction on Autonomous and Adaptive Systems*. 64, 158

Vu, L.H., Hauswirth, M. & Aberer, K. (2005a). QoS-based service selection and ranking with trust and reputation management. In *Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005 Proceedings, Part I*, vol. 3760, 446–483, Springer-Verlag GmbH. 8, 18, 23, 110, 111

Vu, L.H., Hauswirth, M. & Aberer, K. (2005b). Towards P2P-based semantic web service discovery with QoS support. In *Proceedings of Workshop on Business Processes and Services (BPS)*, Nancy, France. 110, 112, 117

Vu, L.H., Hauswirth, M., Porto, F. & Aberer, K. (2006). A search engine for QoS-enabled discovery of Semantic Web services. *International Journal of Business Process Integration and Management*, **1**, 244Ű–255. 110, 112, 117, 121

Vu, L.H., Porto, F., Hauswirth, M. & Aberer, K. (2007). An extensible and personalized approach to QoS-enabled service discovery. In *Eleventh International Database Engineering & Applications Symposium (IDEAS'07), Banff, Canada September 6-8, 2007*, xxx–xxx, IEEE. 110, 112, 117

Vu, L.H., Buchegger, S., Datta, A. & Aberer, K. (2009a). Enabling secure secret sharing in distributed online social networks. In *25th Annual Computer Security Applications Conference*. 110

Vu, L.H., Papaioannou, T.G. & Aberer, K. (2009b). Synergies of different reputation systems: challenges and opportunities. In *2009 World Congress on Privacy, Security, Trust and the Management of e-Business (PST'09), August 25 Ű 27, 2009 Saint John, New Brunswick, Canada*, xxx–xxx, IEEE. 147, 168

Vu, L.H., Papaioannou, T.G. & Aberer, K. (2010). Impacts of trust management and information sharing to adversarial cost in ranking systems. In *4th IFIP WG 11.11 International Conference on Trust Management (IFIPTM'10)*. 168

Wang, Y. & Lin, K.J. (2008). Reputation-oriented trustworthy computing in e-commerce environments. *IEEE Internet Computing*, **12**, 55–59. 11, 15, 16, 148

Wang, Y. & Vassileva, J. (2003). Bayesian network-based trust model. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, 372, IEEE Computer Society, Washington, DC, USA. 25, 48

Wang, Y., Cahill, V., Gray, E., Harris, C. & Liao, L. (2006). Bayesian network based trust management. In *Internation Conference in Autonomic and Trusted Computing*, 246–257. 17, 25, 48

Watts, D.J. & Strogatz, S.H. (1998). Collective dynamics of 'small-world' networks. *Nature*, **393**, 440–442. 138

Whitby, A., Jøsang, A. & Indulska, J. (2005). Filtering out unfair ratings in Bayesian reputation systems. *The Icfain Journal of Management Research*, **4**, 48–64. 24, 25, 48, 65, 114

Wing, J. (2006). Computational thinking. *Communications of the ACM*, **49**, 33–35. 170

Wocjan, P., Janzing, D. & Beth, T. (2002). Required sample size for learning sparse Bayesian networks with many variables. *Arxiv preprint cs.LG/0204052*. 43

Xiong, L. (2005). *Resilient Reputation and Trust Management: Models and Techniques*. Ph.D. thesis, Georgia Institute of Technology. 6, 169

Xiong, L. & Liu, L. (2004). PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.*, **16**, 843–857. 16, 24, 25, 48, 58, 60, 65, 66, 72, 78, 92, 93, 94, 107, 157

Xu, S., Li, X. & Parker, P. (2008). Exploiting social networks for threshold signing: attack-resilience vs. availability. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, 325–336, ACM, New York, NY, USA. 130, 132, 139, 144, 145

Yang, Y.F., Feng, Q.Y., Sun, Y.L. & Dai, Y.F. (2009). Dishonest behaviors in online rating systems: Cyber competition, attack models, and attack generator. *Journal of Computer Science and Technology*, **24**, 855–867. 18, 167

Yin, X., Han, J. & Yu, P. (2007). Truth discovery with multiple conflicting information providers on the web. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1048–1052, ACM, New York, NY, USA. 18

Yu, B. & Singh, M.P. (2000). A social mechanism for reputation management in electronic communities. In *Proceedings of the 4th International Workshop on Cooperative Information Agents (CIA)*, Springer Berlin. 12

Yu, B., Singh, M.P. & Sycara, K. (2004). Developing trust in large-scale peer-to-peer systems. In *Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability*, 1–10. 24, 48, 58

Zhao, S., Lo, V. & Dickey, C.G. (2005). Result verification and trust-based scheduling in peer-to-peer grids. In *Fifth IEEE International Conference on P2P Computing, P2P'05*, 31–38. 19

Le-Hung Vu

Distributed Information Systems Laboratory (LSIR)
Swiss Federal Institute of Technology at Lausanne (EPFL)
BC 142, Station 14, CH-1015 Lausanne, Switzerland

Phone (Office): +41 21 693 7573
Email: lehung.vu@epfl.ch
http://lsirpeople.epfl.ch/lhvu

Research Interests

My research focuses on *soft security* (trust management) mechanisms for e-commerce applications on top of *peer-to-peer* and *social networks*. Currently, I am interested in building *next generation, open reputation-based trust* management systems capable of integrating the information and capabilities from other systems in similar application domains. I also worked on developing Semantic Web service discovery approaches based on quality ratings and user recommendations.

Education

PhD in Computer Science (Advisor: Prof. Karl Aberer), expected May 2010
Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland
Thesis: High quality P2P service provisioning via decentralized trust management.
Corporate training on break-through thinking, project management, team leading, effective communication, and entrepreneurship

Engineering in Computer Science (EPFL Diploma equivalent) 1996-2001
Ho Chi Minh University of Technology, Vietnam
Graduated with high distinction, *ranked 1st* in the class

Work Experience

Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland 2006-2010
Research Assistant- PhD student
Published more than ten papers at good publication venues. Gave public talks or demos at several conferences, project meetings and reviews.
Developed a simulation framework for reputation systems.
Currently working on data access-control in peer-to-peer social networks, a joint research project with T-labs Berlin, Germany and NTU Singapore.

Distributed Information Systems Laboratory, EPFL, Switzerland 2004-2006
Research assistant under support of a fellowship from the Swiss government
Worked in the EU research project DIP, developed solutions and coordinated the implementation a software component to discover Semantic Web services based on their quality properties and user ratings

Ho Chi Minh University of Technology, Vietnam 2001-2004
Assistant lecturer in Computer Science
Taught undergraduate computer science courses, participated in national and university-funded software development projects

Publication

Articles

[1] L.-H. Vu and K. Aberer. Effective usage of computational trust models in rational environments. *in third round review of the ACM Transaction on Autonomous and Adaptive Systems*, 200x.

[2] L.-H. Vu, M. Hauswirth, F. Porto, and K. Aberer. A search engine for QoS-enabled discovery of Semantic Web services. *International Journal of Business Process Integration and Management*, 1(3):244–255, 2006.

Proceedings

[3] L.-H. Vu, T. G. Papaioannou, and K. Aberer. Impacts of trust management and information sharing to adversarial cost in ranking systems. In *4th IFIP WG 11.11 International Conference on Trust Management (IFIPTM'10)*, 2010.

[4] L.-H. Vu, S. Buchegger, A. Datta, and K. Aberer. Enabling secure secret sharing in distributed online social networks. In *25th Annual Computer Security Applications Conference (ACSAC'09)*, 2009 (acceptance rate: 20%).

[5] L.-H. Vu, T. G. Papaioannou, and K. Aberer. Synergies of different reputation systems: challenges and opportunities. In *2009 World Congress on Privacy, Security, Trust and the Management of e-Business (PST'09)*, 2009.

[6] L.-H. Vu and K. Aberer. Towards Probabilistic Estimation of Quality of Online Services. In *2009 IEEE International Conference on Web Services (ICWS 2009)*. IEEE, 2009 (acceptance rate: 16%).

[7] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. PeerSoN: P2P social networking - early experiences and insights. In *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*, 2009.

[8] L.-H. Vu and K. Aberer. Effective usage of computational trust models in rational environments. In *Web Intelligence and Intelligent Agent Technology (WI/IAT'08)*, 2008.

[9] L-.H. Vu, F. Porto, M. Hauswirth, and K. Aberer. An extensible and personalized approach to QoS-enabled service discovery. In *11th International Database Engineering & Applications Symposium (IDEAS'07)*, 2007 (acceptance rate: 33%).

[10] L-.H. Vu and K. Aberer. A probabilistic framework for decentralized management of trust and quality. In *Workshop on Cooperative Information Agents (CIA'07)*, 2007 (runner-up of the best paper award).

[11] L-.H. Vu, M. Hauswirth, and K. Aberer. Towards P2P-based semantic web service discovery with QoS support. In *Workshop on Business Processes and Services (BPS)*, 2005.

[12] L-.H. Vu, M. Hauswirth, and K. Aberer. QoS-based service selection and ranking with trust and reputation management. In *Proceedings of Cooperation Information Systems (CoopIS'05)*, 2005 (acceptance rate: 22%).

Academic Awards

Swiss federal government fellowship for postgraduate study in Switzerland (2004-2006)

University gold medal for being ranked the first among graduates of the Department of IT, Ho Chi Minh University of Technology, Vietnam (2001)

Honorary research award, Ministry of Education and Training, Vietnam (2001)

Second prize in the Young Researchers Competition, Information Technology Track, Ho Chi Minh University of Technology, Vietnam (2001)

Scholarships for high academic achievements granted by the university and leading transnational companies, including Sumitomo, Microsoft, Intel, PSV (1996-2001)

Professional Skills

Extensive experience on reputation systems, data access control, social networks, and on simulation of complex networks.

Very good knowledge and experience in Java, knowledge of Matlab, C/C++, UNIX shell-scripting.

Knowledge of several Java technologies (J2EE, JSP/Servlet, JDBC, JUnit), C/C++ libraries (Qt, MFC, LAM-MPI).

Knowledge of important development methodologies: agile and extreme programming, test-driven development.

Knowledge of most used development tools: Web and application servers (Apache, Tomcat, Orion), DBMS (MySQL, Derby, Oracle), modeling tools (UML/Visio), IDEs (Eclipse, MS Visual Studio, Kdevelop, Qt Designer), versioning tools (SVN, CVS), etc.

Advanced experience working with both UNIX and Windows operating system families.

Languages

Vietnamese (native), English (fluent), French (intermediate), German (beginner)

Professional Activities

Official and/or external reviewer for several journals, conferences, and workshops, including WWWJ, TKDE, ICDE, SIGMOD, VLDB, P2P, WWW, ICWS, WISE (2006-current).

Teaching assistant for Distributed Information Systems, Master course, by Prof. Karl Aberer, winter semester 2007/2008.

Teaching assistant for Théorie et pratique de la programmation, Undergraduate course, by Dr. Monika Lundell and Dr. Michel Schinz, summer semester 2006/2007.

Supervisor of 2 EPFL Master and 3 semster projects (2008-2010) on the topics of peer-to-peer systems, access control on distributed social networks, and Facebook application development.

Other Activities

Coordinator/organizer of the Vietnamese student community in Lausanne, Switzerland.
Admin of the swissviet.com, an online forum that provides various practical information for Vietnamese students in Switzerland.

References

Professor Karl Aberer (karl.aberer@epfl.ch)
Distributed Information Systems Laboratory (LSIR)
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
EPFL-IC-IIF-LSIR, Bâtiment BC, Station 14, CH-1015 Lausanne, Switzerland.
Phone (Office): +41 21 693 4679

Professor Manfred Hauswirth (manfred.hauswirth@deri.org)
Digital Enterprise Research Institute (DERI),
National University of Ireland (NUIG),
IDA Business Park, Lower Dangan, Galway, Ireland.
Phone (Office): +35 3 91 495009

Associate Professor Sonja Buchegger (buc@csc.kth.se)
School of Computer Sciences and Communication (CSC),
Royal Institute of Technology (KTH),
Osquars Backe 2, 4tr., SE-100 44 Stockholm, Sweden.
Phone (Office): +46 8 790 6884