**CHAPTER 1**

# CHAPTER 14: REPUTATION MANAGEMENT SYSTEMS FOR PEER-TO-PEER NETWORKS

F.Qi, H.Shen, H.Chandler, G.Liu and Z.Li

Clemson University, Clemson, South Carolina

## 1.1 INTRODUCTION

In recent years, peer-to-peer (P2P) networks have gained popularity in many large-scale, distributed internet applications. P2P networks enable the sharing of globally scattered computer resources, allowing them to be collectively used in a cooperative manner for different applications such as file sharing [1, 2, 3, 4], instant messaging [5], audio conferencing [6], and distributed computing [7]. Node cooperation is critical to achieving reliable P2P performance but proves challenging since networks feature autonomous nodes without preexisting trust relationships. Additionally, some internal nodes may be compromised, misbehaving, selfish, or even malicious. Then, a critical problem arises: how can a resource requester choose a resource provider that is trustworthy and provides high quality of service (QoS) from among many resource providers?

The main way to address this problem is with reputation management systems. Like the reputation systems in the eBay [8], Amazon [9] and Overstock [10] online auction platforms, reputation systems employed in P2P networks compute and publish global reputation values for each node based on a collection of local ratings from others in order to provide guidance in selecting trustworthy nodes. They thwart the intentions of uncooperative and dishonest nodes and provide incentives for high QoS.

For example, in a P2P file sharing system, a file requester needs to choose a single file provider from many file providers. The requester can refer to the reputation management system to obtain the reputations of the file providers and choose the highest-reputed file provider for high QoS file provision.

Substantial research has been conducted on reputation management systems. These works can be classified into two categories: scalability/accuracy and security. The works on scalability/accuracy describe methods to efficiently collect feedback on past node behaviors, provide node reputation values, and accurately evaluate a node's trustworthiness. The works on security describe methods to avoid malicious behaviors in order to ensure the correct operation and dependability of reputation systems in guiding trustworthy node selection. One particularly bothersome behavior is collusion. A colluding collective is a group of malicious peers who know each other, give each other high ratings, and give all other peers low ratings in an attempt to subvert the system and gain high global reputation values [11]. Reputation systems are generally vulnerable to node collusion [12, 13], and works on collusion resilience describe methods to deter collusion or reduce the adverse effects of collusion on reputation systems. In this chapter, we will focus on discussing these two categories of works.

The rest of this chapter is structured as follows. Section 1.2 introduces works in the above two categories. Section 1.3 presents case studies on three reputation management systems. Section 1.4 discusses open problems with reputation management systems in P2Ps. Finally, Section 1.5 summarizes the chapter.

## 1.2 REPUTATION MANAGEMENT SYSTEMS

### 1.2.1 Approaches for Scalability and Accuracy

In a P2P network, millions of nodes are scattered across geographically distributed areas and enter or leave the system continuously and unpredictably. The large scale, wide geographic distribution, and dynamism of these networks pose a challenge to achieving high scalability in reputation management. Also, the accuracy of calculated global node reputation values determines the effectiveness of a reputation system in discouraging uncooperative and dishonest behaviors and encouraging cooperative behaviors. This section presents the techniques that enable a reputation system to scale to serve a large number of nodes in terms of feedback collection efficiency and overhead, and to achieve high accuracy in node trustworthiness reflection.

***EigenTrust*** EigenTrust [11] can minimize the impact of malicious peers on the performance of a P2P network. In this system, the global reputation of a peer is calculated by its received local trust values from other peers and weighted by the global reputations of the peers. Specifically, the system computes a global trust value for a peer by calculating the left principal eigenvector of a matrix of normalized local trust values, thus taking into consideration the entire system's history with each single peer. EigenTrust carries out these computations in a scalable and distributed manner; all peers in the network participate in computing global reputation values node-symmetrically, with minimal overhead on the network. Furthermore, EigenTrust ensures the security of the computations and minimizes the probability that malicious peers in the system lie for their own benefit. Peers that provide material deemed inappropriate by the users of a P2P network are accurately identified by EigenTrust and effectively isolated from the network. This system is highly effective in decreasing the number of unsatisfactory downloads, even when up to 70% of the peers in the network form a malicious collective in an attempt to subvert the system. In P2P simulations, using reputation values to bias peers against certain providers has shown to reduce the number of inauthentic files in the network under a variety of threat scenarios. Furthermore, rewarding highly reputable peers with better quality of service induces non-malicious peers to share more files and to self-police their own file repository for inauthentic files.

***PeerTrust*** PeerTrust [14] is a dynamic P2P trust model for quantifying and assessing the trustworthiness of peers in P2P e-commerce communities. A unique characteristic of the trust model is the identification of five important factors for evaluating the trustworthiness of a peer in an evolving P2P e-commerce community, as listed below.

1. The feedback a peer receives from other peers.
2. The feedback scope, such as the total number of transactions that a peer has conducted with other peers.
3. The credibility of the feedback sources.
4. The transaction context factor for distinguishing mission-critical transactions from less critical or noncritical ones.
5. The community context factor for addressing community-related characteristics and vulnerabilities.

PeerTrust defines a general trust metric that combines the parameters of the above factors, and implements the trust model in a decentralized P2P environment. It can significantly reduce common security threats in P2P environments, such as man-in-the-middle attacks, compromised peers, and the distribution of tampered-with information.

***TrustGuard*** TrustGuard [15] is a highly dependable reputation-based trust building framework with a storage service built on top of PeerTrust. The TrustGuard framework is equipped with several safeguards that are critical for minimizing the potential threats and vulnerabilities in the reputation system itself. It guards against strategic oscillations, detects fake transactions, and filters out dishonest feedback. First, TrustGuard incorporates the historical reputations and behavioral fluctuations of a node into the estimation of its trustworthiness, guaranteeing that reputation is built gradually but drops quickly if a node starts to behave maliciously. Second, TrustGuard has a feedback admission control mechanism to ensure that only transactions with secure proofs can be used to file feedback; this prevents malicious nodes from misusing the system by flooding feedback. Third, TrustGuard has an effective mechanism to rate the feedback credibility of nodes and discount dishonest feedback in order to filter out dishonest feedback when computing the reputation-based trust of a node (including the feedback filed by malicious nodes through collusion). TrustGuard's approach can efficiently and effectively secure a large-scale distributed reputation system, making it more dependable than other existing reputation-based trust systems.

***FuzzyTrust*** FuzzyTrust [16] is a prototype P2P reputation system that helps establish mutual trust among strangers in P2P transaction applications. The authors first analyzed auction-based transaction trace data from eBay to sort out client behavioral characteristics, and then proposed FuzzyTrust based on the data analysis. The system uses fuzzy logic inference rules to calculate local trust scores and to aggregate global reputation. It benefits from the distinct advantages of fuzzy inferences, which can effectively handle imprecise or uncertain linguistic terms collected from peers. Furthermore, the system uses a distributed hash table (DHT) overlay network to perform fast and secure reputation dissemination among peers. Experimental results show that FuzzyTrust is identifies malicious peers effectively and has a low message overhead in the global reputation aggregation process.

***Reputation Systems for Pollution Avoidance*** In content pollution in P2P file-sharing systems, polluters camouflage polluted files and share them with other users; unsuspecting users then download these files, wasting bandwidth and CPU resources. Furthermore, polluted files are often left in the shared folders of normal users, thus rapidly spreading the files through the system. Credence [17] and Scrubber [18] are reputation systems that fight content pollution.

Credence is a decentralized distributed object reputation system in which users assign reputations to the objects they download with regard to their authenticity. It is based on a distributed vote gathering protocol for disseminating object reputations in the network and on a correlation scheme that more heavily weights votes from like-minded peers. Credence also periodically runs a gossip protocol where each peer randomly selects another peer to retrieve its correlation coefficients. Transitive correlations are computed by multiplying retrieved correlation coefficients by the local weights given to the contacted peers.

Scrubber is a distributed and decentralized peer reputation system that quickly identifies and severely punishes active polluters, but also includes inherent incentives for peer rehabilitation by giving passive polluters (i.e., peers that share polluted content by negligence) an incentive to remove polluted content they have downloaded. Scrubber has a distributed and decentralized architecture, thus simplifying deployment. Compared with Credence, Scrubber converges much faster to a competitive maximum efficiency, quickly reducing the fraction of daily downloads that are polluted objects to less than 8%, as long as at least 25% of the peers react to punishment by deleting their polluted objects. Otherwise, Credence somewhat outperforms Scrubber in the long run.

Costa and Almeida proposed a Hybrid Reputation System [19] that is a hybrid peer and object reputation system combining the benefits of both Scrubber and Credence. Despite a quick convergence, Scrubber is not always able to clean polluted objects shared by peers that only occasionally upload them. Credence, on the other hand, converges much more slowly but is eventually able to isolate all polluted objects. The hybrid system combines the benefits of both Scrubber and Credence by building a mechanism for peers to vote on the authenticity of objects into Scrubber. The hybrid system has two key components: the object reputation and the peer reputation. The peer and object reputations are evaluated based on the assumption that a peer well-reputed as a content provider (or voter) is an honest reputation reporter. Like Scrubber, the hybrid system must take the network opinion into account by applying quick punishments to polluters to be a viable solution for large-scale P2P systems, where the frequency of interaction between the same pair of peers is typically low. Memory requirements are roughly equal for all three systems, and most memory is used for the storage of local and remote opinions of peers. The main benefits are (1) the hybrid system converges to a maximum efficiency much faster than Credence and Scrubber, even under collusion and Sybil attacks [20], (2) the hybrid system is less sensitive to parameter settings than Scrubber, providing cost effectiveness for various configurations, and (3) the hybrid system is able to restrain pollution dissemination even in very un-cooperative and unreliable communities, despite depending greatly on user cooperation and reliable feedback.

***GossipTrust*** GossipTrust [21] is a scalable, robust, and secure reputation management system specifically designed for unstructured P2P networks. This system leverages a gossip-based protocol to aggregate global reputation scores; specifically, each peer randomly contacts others and exchanges reputation data periodically. Gossip-based protocols do not require any error recovery mechanism, and thus enjoy simplicity and moderate overhead when compared with optimal deterministic protocols [22] such as the construction of data dissemination trees. GossipTrust is built around a fast reputation aggregation module with enhanced security support that strengthens the robustness of the gossip protocol under disturbances from malicious peers. The system has a novel data management scheme to answer reputation queries and to store reputation data with low overhead. Identity-based cryptography is applied to ensure the confidentiality, integrity, and authenticity of the exchanged reputation data without using certified public keys or preshared secret keys.

GossipTrust was improved [23] with higher speed and accuracy in aggregating local trust scores into global reputation ranks. The improved GossipTrust enables peers to compute global reputation scores in a fully distributed, secure, scalable, and robust fashion. Simulation results show that the system scales well with increasing network size. The system can also tolerate link failures and peer collusion. The technical innovations of this improved reputation system are summarized in four aspects:

1. Fast gossip-based reputation aggregation algorithms with small aggregation error: the $O(\log_2 n)$ time complexity, where $n$ is the number of nodes in the network, makes the gossip search as attractive as DHT-based table lookup.

2. Efficient reputation storage using Bloom filters with low false-positive error: even for a network of one million nodes, the memory required to rank nodes by reputation is just 512 KB per node, and the false-positive error is at most 15%.

3. Limited network traffic overhead in gossip message spreading: the total network traffic increase of $O(n \log_2 n)$ is low compared to multicast or broadcast approaches.

4. Combating peer collusion by using power nodes dynamically: Gossip-Trust leverages the rank of all nodes in terms of their relative standing in the global reputation vector. The effects of peer collusion are minimized due to each colluding peer's low rank.

***PErsonalized Trust*** PErsonalized Trust(PET) [24] is a personalized trust model in the context of economic-based solutions for P2P resource sharing. The trust model consists of two parts: reputation evaluation and risk evaluation. Reputation is the accumulative assessment of a long-term behavior, while risk evaluation is the assessment of a short-term behavior. Risk is employed to deal with dramatic behavior changes in peers. PET novelly models risk as an opinion of short-term trustworthiness and combines this with a traditional reputation evaluation to derive trustworthiness. In this model, recommendations play a moderate role as one of the many factors from which local trustworthiness values are derived. PET only takes into account a peer's behavior within an employed risk window; as the window shifts forward, the risk value reflects the fresh statistics of the peer's recent behaviors. The risk model is very important in PET since risk evaluation can combat malicious recommendations. The authors' observations are summarized below.

1. Giving risk a high weight more effectively improves the performance of the model in such measures as sensitivity, effectiveness, hit ratio, and applicability when more peers in the community are malicious or uncooperative.

2. A small risk window size is helpful in improving the sensitivity of the model when the weight of the risk is high.

3. Giving recommendations a low weight improves the performance of the model while maintaining resistance to malicious recommendations.

The PET model is promising for resource sharing in P2P networks with large numbers of dynamic peers, uncooperative peers, and malicious recommenders.

***H-Trust*** Peer-to-Peer Desktop Grid (P2PDG) has emerged as a pervasive cyber-infrastructure tackling many large-scale applications with high impact, such as SETI@Home and DNA@Home. In a P2PDG environment, users run large-scale and cooperative computational applications, and most computing jobs are accomplished by work groups. However, nearly all existing reputation systems do not consider issues with group reputation for collaborative services and resource sharing. In addition, most of the current reputation schemes must decide between low network overhead or high accuracy in reputation aggregation. To address these issues, a robust and lightweight group trust management system was proposed, called H-Trust [25], which was inspired by the H-index aggregation approach [26]. Leveraging the robustness of the H-index algorithm under incomplete and uncertain circumstances, H-Trust offers a robust reputation evaluation mechanism for both individual and group trusts with minimal communication and computation overhead. Users in this system only store information they can explicitly use for their own benefit. H-Trust further considers spatial and temporal information to update and adapt trust scores for individuals and groups. The H-Trust reputation aggregation scheme is implemented in five phases.

1. The trust recording phase records past service information in a trust history table, which is maintained in the DHT-based overlay network.

2. In the local trust evaluation phase, a local trust score is calculated by a local trust manager using a weighted reputation aggregation algorithm.

3. The trust query phase is required when trust information is not available locally. The credibility of the responses and the H-Index aggregation are proposed in this phase to yield an individual's reputation.

4. The spatial-temporal update phase is activated periodically to renew local trust scores and credibility factors.

5. The group reputation evaluation phase aggregates a group reputation using the H-Trust algorithm.

***A Trust Inference System*** Lee *et al.* [27] proposed a distributed scheme for trust inference in the context of the NICE system, which is a platform for implementing cooperative applications over the Internet. The inferred trust values represent how likely a user considers other users to be cooperative and are used to price resources in the NICE system.

The trust inference system aims to classify all users as either cooperative or uncooperative with no errors. In this scheme, for each transaction in the system, each involved user produces a signed statement (called a cookie) about the quality of the transaction. The scheme infers trust using a directed graph called a trust graph. The vertices in the graph correspond exactly to the users in the system. There is an edge directed from user A to user B if and only if user B holds a cookie from user A. The value of the $A--B$ edge denotes how much user $A$ trusts user $B$, and depends on the set of user $A$'s cookies held by user $B$.

Given a path $A_0 \rightarrow A_1 \rightarrow A_k$ in the trust graph, $A_0$ could infer a number of plausible trust values for $A_k$, including the minimum value of any edge on the path or the product of the trust values along the path; these inferred trust values are called the strength of the $A_0 \rightarrow A_k$ path. Assume node A has access to the trust graph, and wants to infer a trust value for node B. Node A can use (1) strongest path, which takes the minimum trust value on the path as the trust value for B, or (2) a weighted sum of strongest disjoint paths. Users locate trust information about other users in a distributed manner. Each user stores a set of signed cookies that it receives as a result of previous transactions. Suppose node A wants to use some resources at node B. There are two possibilities: either A already has cookies from B, or nodes A and B have not had any transactions yet. For the case in which A already has cookies from B, A presents these to B. Node B can verify that these are indeed its cookies since it has signed them. From the cookies, node B can compute a trust value for A. When A has no cookies from B, A initiates a search for B's cookies at nodes from whom A holds cookies. After the search is over, A presents B with a union of directed paths which all start at B and end at A, which correspond exactly to the union of directed edges on the trust graph used for the previously described centralized trust inference. Thus, B can infer a trust value for A.

Through the above scheme, the trust inference system lets benign nodes find each other quickly and efficiently, and prevents malicious nodes and cliques from breaking up cooperating groups by spreading misinformation to benign nodes; thus, the system achieves its goal of classifying all users as either cooperative or uncooperative.

***A Reputation-Based Trust Management System*** In the reputation-based trust management system proposed in [28], each peer maintains a trust vector for every other peer it has dealt with in the past. Trust vectors are constant length and binary. A 1 bit represents an honest transaction, and a 0 bit represents a dishonest one. Based on the trust vector, a peer calculates trust and distrust ratings for other peers. The trust query process is similar to the file query process, except that the subject of the query is a peer about whom trust information is inquired. The responses are sorted and weighted by the credibility ratings of the responders. Credibility ratings are derived from the credibility vectors maintained by the local peer, which are similar to the trust vectors: a 0 in a credibility vector shows a failed judgment from that peer in the past, and a 1 shows a successful judgment. The threshold specifies the number of responses to be evaluated for each trust query. The queried trust/distrust rating is the average of the evaluated trust ratings weighted by the credibility of their senders. This trust management system features:

1. The separate treatment of distrust ratings: handling distrust ratings separately means that a dishonest dealing cannot be easily erased by a few honest transactions, thus closely modeling real-life trust relationships in which a single dishonest transaction in someone's history is a more significant indicator than several honest transactions.

2. Temporal adaptivity consideration, i.e., the ability to respond rapidly to changing behavioral patterns: the trust rating design utilizing binary vectors is an efficient exponential aging scheme with an aging factor of 0.5. Moreover, implementing the aging scheme by fixed-length registers rather than floating point arithmetic has the desirable feature of

enabling peers to cleanse their history by doing a reasonable amount of community service after a bad deed.

3. The use of a credibility rating system separate from trust ratings: the main risk of using trust ratings for credibility evaluation comes from coordinated attacks where some malicious peers do as much faithful public service as they can to build a strong reputation, and then use their credibility for supporting others who spread malicious content. Having separate trust and credibility rating systems precludes such attacks.

### 1.2.2 Approaches for Security

An important challenge in managing trust relationships is designing a protocol to secure the placement and access of trust ratings. Specifically, a reputation system should ensure the following [29]:

1. Security. Due to decentralized management of trust relationships, the trust rating of a peer is stored at other peers in the network; it is critical that these trust hosting peers are protected from targeted attacks.

2. Reliability. It is important that anybody querying for a trust value gets the true trust value, despite the presence of various malicious users.

3. Accountability. In peer-review based trust systems, it is important that peers are accountable for the feedback they provide about other peers. Any malicious peer trying to manipulate trust ratings should be identifiable.

In this section, we present the methods proposed to achieve some of the above properties.

***Maze P2P File Sharing System*** Of particular concern in reputation and trust systems is collusion, i.e., multiple nodes working together to game the system. Collusion subverts any strategy in which everyone in the system agrees on the reputation of a player (objective reputation). The effect of collusion is magnified in systems with zero-cost identities, where users can create fake identities that report false statements [30].

Maze [12] is a popular Napster-like P2P network designed, implemented, and deployed by an academic research team at Peking University, Beijing, China. The authors searched for the existence of a colluding behavior by examining the complete user logs of the entire system, and used a set of collusion detectors to identify several major collusion patterns.

1. Detector 1 (Repetition detector): large amounts of upload traffic with repeated content.

2. Detector 2 (Pair-wise detector): large amounts of mutual upload traffic between a pair of nodes compared to total uploads.

3. Detector 3 (Spam account detector): high peer to machine ratios indicating spam account collusion.

4. Detector 4 (Traffic concentration detector): exceptionally high traffic concentration degrees, which is the ratio of a peer's highest upload traffic to a single machine to his total upload traffic.

***Stamp Trading Protocol*** Reputation and payment protocols are two methods of introducing cooperation incentives to nodes in P2P networks. In reputation protocols, nodes with low reputations will find it difficult to obtain service in the network. A reputation protocol can ensure that nodes need to successfully contribute to the network in order to have a high reputation. In payment protocols, nodes receive credits only by successfully providing service to other nodes. Those that do not provide services cannot gain the credits that they need to buy services. The stamp trading protocol [31] is a natural generalization of both reputation and payment protocols. In the protocol, nodes issue or trade personalized stamps with their neighbors, which can later be redeemed for services at the issuing nodes. In order to obtain service from node $i$, nodes need to present stamps originally issued by node $i$. A node can trade either its own stamps or those it has received from other nodes. By relating the exchange rate of stamps to their issuers' behavior, it is in a node's interest to get into a position where it is able to obtain sufficient stamps to do what it wants. The exact nature of the incentives arises in the method used to determine the stamp exchange rates.

A stamp trading scheme is a stamp trading protocol along with a method for valuing the stamps. The stamps that a node has in circulation represent the amount of service to which it has committed itself. A node's credit is the total value of stamps it has on hand (stamps not issued by itself plus the total value of stamps that it has yet to issue). Because nodes can give stamps away to neighboring nodes, the total credit in the network equals the total value of stamps in circulation (stamps issued by a node are held on hand by others in the network). A stamp trading scheme is token-compatible if the total credit (value of stamps in circulation) in the network is bounded. This fits the notion of a payment protocol, where tokens cannot be forged or minted so that the economy is bounded. Additionally, a scheme is trust-compatible if failure by a node to successfully redeem a stamp never increases its credit, i.e., stamp value is monotonically decreasing with an increasing number of failures. This fits the notion of a reputation protocol, where nodes cannot gain trust by misbehaving.

***Reciprocative Decision Function*** Feldman *et al.* [30] modeled the P2P network using the Generalized Prisoner's Dilemma (GPD) and proposed the Reciprocative decision function as the basis of a family of incentives techniques including discriminating server selection, maxflow-based subjective reputation, and adaptive stranger policies. Specifically, the authors used GPD to capture the essential tension between individual and overall network utility; asymmetric payoff matrices to allow asymmetric transactions between peers; and a learning-based [32] population dynamic model to specify the behavior of individual peers, which can be changed continuously. The proposed family of scalable and robust incentive techniques include:

1. Discriminating Server Selection: cooperation requires familiarity between entities either directly or indirectly. However, the large population and high turnover of P2P systems make it less likely that repeat interactions will occur with a familiar entity. If each peer keeps a private history of the actions of other peers toward itself and uses discriminating server selection, the Reciprocative Decision Function can scale to large populations and moderate levels of turnover.

2. Shared History: scaling to higher turnover and mitigating asymmetry of interest requires shared history. For example, take three nodes A, B, and C, where C has been served by A and has served B. With shared history, B can be familiar with A's service to C and would thus be willing to serve A. With only private history, B would not know that A has served C, so B would not be willing to serve A. Shared history results in a higher level of cooperation than private history. The cost of shared history is a distributed infrastructure (e.g., distributed hash table-based storage) to store the history.

3. Maxflow-based Subjective Reputation: shared history creates the possibility of collusion. For example, C can falsely claim that A served him, thus deceiving B into providing service. A maxflow-based algorithm that computes reputation subjectively promotes cooperation de-

spite collusion; using this algorithm in the previous example, B would only believe C if C had already provided service to B.

4. Adaptive Stranger Policy: zero-cost identities allow non-cooperating peers to escape the consequences of their behaviors by switching to a new identity. If reciprocative peers treat strangers (peers with no history) using a policy that adapts to the behaviors of previous strangers, peers have little incentive to switch to new identities.

5. Short-term History: history also creates the possibility that a previously well-behaved peer with a good reputation will turn malicious and use its good reputation to exploit other peers. The peer could be making a strategic decision or someone may have hijacked its identity (e.g., by compromising its host). Long-term history exacerbates this problem by allowing peers with many previous transactions to exploit that history for many new transactions; short-term history prevents malicious nodes from disrupting cooperation.

***XRep and $X^2$Rep*** XRep [33] is a reputation-based trust management system designed to reduce the number of malicious or low quality resources distributed in a Gnutella file-sharing network. Associating reputations with servents becomes a difficult problem for an anonymous P2P environment, where resource providers are identified by a pseudonym and an IP address. To overcome the limitations of servent-only based methods, the XRep protocol uses the combined reputations of servents and resources. Servent reputations are associated with a tamper-resistant servent identifier. Resource reputations are tightly coupled to the resources' content via their digest, thus preventing their forging on the part of malicious peers. Reputations are cooperatively managed via a distributed polling algorithm in order to reflect the community's view of the potential risk involved with the download and use of a resource. Each servent maintains information on its own experiences with resources and other servents, and can share such experiences with others upon request. The XRep protocol also improves the global security and quality of content distribution within P2P networks. It protects P2P networks against most known attacks such as self-replication, man-in-the-middle, pseudospoofing, ID stealth, and shilling.

Trust semantics specify the model for the evaluation of trust through the computation of gathered reputation information. $X^2$Rep [34] enhances the trust semantics of the XRep Protocol. $X^2$Rep gives peers more expressive power to express their opinion about resources that they have downloaded and resource providers.

Ensuring the reliability of gathered reputation information is a major challenge to the development of a reputation system. In particular, it is vital that any "vote spoofing" activity is as difficult or expensive as possible for malicious agents. The XRep protocol uses a complex process of challenge and response messages to ensure that a vote is supplied by a real peer. $X^2$Rep eliminates this complexity by employing an extensive vote generation and evaluation system that makes use of voter credibility information to help an evaluating peer determine the trustworthiness of a vote through the evaluation of the voter's previous voting activity. The $X^2$Rep reputation system provides safeguards against threats posed by the collusion of malicious peers, achieving its security goal whilst reducing communication overhead.

***Sorcery*** Sorcery [35] aims to detect the deceptive behavior of peers in P2P networks. It is a challenge-response mechanism based on the notion that the participant with dominant information in an interaction can detect whether the other participant is telling a lie. Here challenge denotes a query about votes for some content, and response denotes the response messages to answer the challenge. Sorcery encompasses three key techniques to detect and punish the deceivers in P2P content sharing systems.

1. Social network mechanism. So that each client has dominant information, Sorcery introduces a social network into the P2P content sharing system; thus, each client can establish his own friend relationships. These friends share their own information (e.g., content and votes) with the client, and the friend information of the client is confidential to other peers in the system.

2. Challenge-response mechanism. Sorcery clients utilize the voting histories of their friends to test the voting history of the content provider and judge whether the content provider is a deceiver or not based on the correctness of his response.

3. Punishment mechanism. Sorcery clients rank each search result based on the honesty of the content providers; therefore, the probability of impact brought by deceivers is reduced.

Sorcery can effectively address the problem of deceptive behavior based on the confidential information extracted from social networks. However, some other types of attacks can also be mounted against Sorcery, such as man-in-the-middle attacks, Sybil attacks, and DoS attacks.

***P2PRep*** In P2PRep [36], servents can keep track of and share the reputations of their peers. Reputation sharing is based on a distributed polling algorithm by which resource requesters can access the reliability of perspective providers before initiating a download in the P2P network. P2PRep allows a servent $p$ to inquire about the reputation of providers by polling its peers before deciding from where to download a file. After receiving the responses, $p$ selects a servent (or a set of servents) based on the quality of the provider and its own past experience. Then, $p$ polls its peers by broadcasting a message requesting their opinion about the selected servents. All peers can respond to the poll with their opinions about the reputation of each servent. The poller $p$ can use the opinions expressed by these voters to make its decision. There are two variations of this approach. In the first variation, called basic polling, the servents responding to the poll do not provide their *servent˙id*. In the second variation, called enhanced polling, voters provide their *servent˙id*, which

$p$ can use to weight the votes received ($p$ can judge some voters as being more credible than others).

This approach is complicated by the need to prevent exposure of polling to security violations by malicious peers. In particular, both the authenticity of servents and the quality of the poll needs to be ensured. Ensuring the quality of the poll means ensuring the integrity of each single vote (e.g., detecting modifications to votes in transit) and ruling out the possibility of dummy votes expressed by servents acting as a clique under the control of a single malicious party. To this end, P2PRep has a *suspects identification* procedure that tries to reduce the impact of forged votes. This procedure relies on computing clusters of voters whose common characteristics suggest that they may have been created by a single, possibly malicious user.

***Honest Players*** In general, existing trust schemes are only effective when applied to honest players who act with consistency, as opposed to adversaries who can behave arbitrarily. The work in [37] investigated the modeling of honest entities in decentralized systems, and built a statistical model for the transaction histories of honest players. This statistical model serves as a profiling tool to identify suspicious entities. It is combined with existing trust schemes to ensure that the schemes are applied to entities whose transaction records are consistent with the statistical model. A two-phase approach is used to integrate the modeling of honest players with trust functions. In the first phase, the transaction history of an entity is examined. If the entity follows the model of honest players, then trust functions will be applied to further determine trustworthiness. Those who do not pass the first phase may either be discarded as untrustworthy (as they appear to manipulate the reputation system) or selected for further examination. This approach limits the manipulation capability of adversaries, and thus can improve the quality of reputation-based trust assessment. The contributions of this work are detailed as below:

1. A statistical model of the behavior of honest players. Specifically, the number of good transactions (those offering satisfactory services and receiving positive feedback accordingly) of an honest player is considered to be a random variable $x$. The work shows that if an entity's behavior is consistent and not affected by other factors, then $x$ follows a binomial distribution $B(n, p)$, where $n$ is the number of transactions a party conducted during a period of time, and $p$ is the percentage of good transactions among these $n$ transactions.
2. An algorithm determines with high confidence whether a party follows the behavior of honest players given the transaction history of a party.
3. An extended statistical model of behavior resistant to collusion and false feedback.

***SFTrust*** Most of the trust models use a single trust metric, which cannot reflect the practical trust values of peers effectively. SFTrust [38] is a decentralized and dependable trust model in unstructured P2P networks using service trust and feedback trust. In SFTrust, each peer has two trust values: service trust and feedback trust. Service trust represents whether the peer can supply high quality service, while feedback trust denotes whether the peer can give equitable recommendations to other peers. The two trust metrics are independent. For example, a peer with a high service trust and a low feedback trust is able to supply high quality service, but lies when giving feedback so as to indirectly improve its trust values by debasing other peers.

SFTrust has three major building modules: a trust storage module, a trust computing module and a trust update module. Specifically, when peer $i$ requests services from peer $j$, it calculates the service trust of $j$ to decide whether peer $j$ is a trustable provider. First, $i$ checks its service trust table to find the direct trust of $j$, i.e., the trust value based on the direct experiences of $i$. Second, $i$ aggregates the weighted trust values about $j$ from its neighbor peers with the recommendation management module. Third, $i$ uses the trust computing module to compute $j$'s service trust by integrating direct trust and recommending trust. Last but not least, $i$ decides whether to obtain services from $j$ while simultaneously updating the feedback trust of recommending peers in its feedback trust record module.

In unstructured P2P trust models, the topology is not strictly defined, and there is no relationship between trust storage and topology. Each peer has a set of neighbors that are chosen upon joining the system. Thus, malicious peers can form a spiteful group to destroy the system to some extent, where they besmirch the reputations of good peers or help malicious peers obtain high reputations. Because of the large quantity of peers, general punishment mechanisms cannot accurately identify collusive peers. In SFTrust, the topology adaption protocol establishes neighborships in such a way that malicious nodes are prevented from easily forming collusive groups. SFTrust also includes counter measures to resist other attacks such as on-off attacks [39], free riding [14], Sybil attacks, and newcomer attacks [40].

***TrustMe*** TrustMe [29] is an anonymous and secure protocol for maintaining and accessing trust rating information. TrustMe uses public key cryptography schemes to provide security and is resistant to various attacks. It is characterized by its support for mutual anonymity in managing peer trust relationships; both the peers who access the trust ratings and the peers who store the trust ratings remain anonymous. This protects the peers who store trust ratings from targeted attacks. TrustMe mainly addresses two problems: (1) where should the trust value of a peer be stored? And (2) how can other peers' trust values be securely accessed? TrustMe broadly functions in the following manner. Each peer is equipped with several public-private key pairs. The trust values of a peer, say peer B, are randomly assigned to another peer (Trust-Holding Agent (THA) peer) in the network. This assignment is done by the bootstrap server in such a way that the trust holding responsibilities are equally distributed amongst the participating peers. This assignment is unknown to all peers, including peer B. All communication with the THA peer is carried out using a special key that indicates its knowledge of the trust value of peer B. One peer interested in querying for the trust value of peer B, say peer A, can broadcast a trust query for peer B. The THA peer replies with the trust value along with some other information. Depending upon the trust value, peer A can decide to interact with peer B or not. Also, after an interaction, peer A can securely file a report (after giving adequate proof of the interaction) on peer B indicating peer A's new trust value for peer B. Then, the THA peer can modify the trust rating of peer B. To provide security, reliability, and accountability, TrustMe uses tested, effective public key cryptography mechanisms.

***Pseudo Trust*** Most trust models in P2P networks are identity-based, which means that in order for one peer to trust another, it needs to know the other peer's identity. Hence, there exists an inherent tradeoff between trust and anonymity. Since currently no P2P protocol provides complete mutual anonymity as well as authentication and trust management, the Pseudo Trust (PT) protocol [41] was proposed. It is a zero-knowledge authentication scheme, in which each peer generates an unforgeable and verifiable pseudonym using a one-way hash function so that peers can be authenticated without leaking sensitive information. Peers construct anonymous onion paths and find tail nodes based on the APFS protocol [42]. PT has five key components, including (1) pseudo identity generation and issuance, (2) new peer initialization, (3) authentication and session key exchange, (4) file delivery, and (5) trust and reputation management. PT can effectively defend against impersonation, replay attacks, man-in-the-middle attacks, collaborated attacks, and denial of service attacks. With the help of PT, most existing identity-based trust management schemes become applicable in mutually anonymous P2P systems. The design strengths of PT include (1) no need for a centralized trusted party, (2) high scalability and security, (3) low traffic and cryptography processing overhead, and (4) man-in-the-middle attack resistance.

***Reliable Rating Aggregation System*** The reliability of online rating systems largely depends on whether unfair ratings and dishonest raters can be detected and removed. Dealing with unfair and dishonest ratings in online feedback-based rating systems has been recognized as an important problem. The lack of realistic attack behavior models and unfair rating data from real human users has become an obstacle toward developing reliable rating systems. To solve this problem, Feng *et al.* [43] designed and launched a rating challenge to collect unfair rating data from real human users. In order to broaden the scope of the data collection, they also developed a Reliable Rating Aggregation System [43], a signal-based unfair rating detection system [43]. The detection system not only outperforms existing schemes but also encourages creative attacks from the participants in the rating challenge. The process of the Reliable Rating Aggregation System contains four steps.

1. Raw ratings are analyzed. Four analysis methods (arrival rate detection, model change detection, histogram detection and mean change detection) are applied independently.

2. The outcomes of the four detectors are combined to detect the time intervals in which unfair ratings are highly likely. Additionally, the suspicious rating detection module can mark some specific ratings as suspicious.

3. A trust manager, which is a simplification of the generic framework of trust establishment proposed in [44], determines how much individual raters can be trusted.

4. Highly suspicious ratings are removed from the raw ratings by a ratings filter. Then, the ratings are combined by the rating aggregation algorithm.

***Reputation-based Fines in Electronic Markets*** The effectiveness of online feedback mechanisms for rating the performance of providers in electronic markets can be hurt by the submission of dishonest ratings. Papaioannou and Stamoulis [45] dealt with ways to elicit honest ratings in a competitive electronic market where each participant can occasionally act as both provider and client. The authors assumed that each service provision is rated by both parties involved; only when the ratings agree are they included in the calculation of reputation for the provider's performance. The authors first studied the effectiveness of an incentive mechanism as a single-shot game. That is, upon evidence of lying (i.e. disagreement between submitted feedback), there are fixed fines to both parties that differ for the provider and the client. The authors proved that the submission of honest feedback can be a stable equilibrium for the whole market under certain initial system conditions. Then, the authors refined the game model for repeated transactions and calculated proper reputation-based fines for lying. These fines make the submission of honest feedback a stable Nash equilibrium of the repeated game and reduce social losses due to unfair punishments.

***Decentralized Recommendation Chains*** The absence of a central authority in P2P networks poses unique challenges for reputation management in the network, the most important of which is availability of reputation data. Dewan and Dasgupta [46] presented a cryptographic protocol for ensuring the secure and timely availability of a peer's reputation data to other peers at low cost. The cryptographic protocol is coupled with self-certification and cryptographic mechanisms for identity management and Sybil attack resistance.

All peers in the P2P network are identified by identity certificates (aka identities). The reputation of a given peer is attached to its identity. The identity certificates are generated using self-certification, and all peers maintain their own (and hence trusted) certificate authority which issues the identity certificate(s) to the peer. Each peer owns the reputation information pertaining to its past transactions with other peers in the network and stores the information locally. The main contributions of this work include:

1. A self-certification-based identity system protected by cryptographically blind identity mechanisms. This reduces the threat of Sybil attacks by

binding the network identity of a peer to his or her real-life identity while still providing anonymity.

2. A lightweight and simple reputation model. This reduces the number of malicious transactions and consumes less bandwidth per transaction.

3. An attack resistant cryptographic protocol for the generation of authentic global reputation information. The global reputation data are protected against any malicious modification by a third party peer and are immune to any malicious modifications by their owner.

***A Robust Reputation System*** Buchegger and Boudec [47] proposed a robust reputation system to cope with the spread of false reputation ratings (i.e., false accusations or false praise). In the proposed system, everyone maintains a reputation rating and a trust rating about everyone else who they care about. For example, node $i$ maintains two ratings about node $j$. The reputation rating represents the opinion formed by node $i$ about node $j$'s behavior as an actor in the base system (e.g., whether node $j$ provides the correct files in a P2P file-sharing system). The trust rating represents node $i$'s opinion about how honest node $j$ is as an actor in the reputation system (i.e., whether the reported first-hand information summaries published by node $j$ are likely to be true). From time to time first-hand reputation information is exchanged with others; using a modified Bayesian approach, only second-hand reputation information that is not incompatible with the current reputation rating is accepted. Thus, reputation ratings are slightly modified by accepted information. Trust ratings are updated based on the compatibility of second-hand reputation information with prior reputation ratings. Data is entirely distributed; a node's reputation and trust is the collection of ratings maintained by others. Every node uses its rating to periodically classify other nodes according to two criteria: 1) normal/misbehaving, and 2) trustworthy/untrustworthy. Both classifications are performed using the Bayesian approach. Re-evaluation and reputation fading are further employed to enable redemption and prevent the sudden exploitation of good reputations built over time.

***A Fine-Grained Reputation System*** Zhang and Fang [48] found three problems in previous reputation systems: (1) a binary QoS differentiation method that classifies a service as either good or bad without any interim state, thus limiting the potential for use by P2P applications in which servers have diverse capabilities and clients have various QoS demands; (2) no strong incentives designed to stimulate honest participation in the reputation system; (3) failure to protect the privacy of references, which is important for obtaining honest feedback. To address these problems, the authors proposed a fine-grained reputation system to support reliable service selection in P2P networks with the following properties:

1. QoS aware. The authors proposed a Dirichlet reputation engine based on multivariate Bayesian inference [49]. Firmly rooted in statistics, the

reputation engine can satisfy the diverse QoS requirements of individual nodes by means of a fine-grained QoS differentiation method.

2. Incentive aware. Honest participation in the reputation system is motivated by charging users who inquire about others' reputations and rewarding those who provide honest feedback.

3. Socially aware. The concept of social groups is incorporated into the reputation system design as a reliable means of soliciting honest feedback and alleviating the cold-start problem. This design is motivated by the sociological fact that people tend to contribute to their associated social groups.

4. Application independent. Unlike many previous solutions that were designed for a concrete P2P application, this reputation system can simultaneously serve unlimited P2P applications of different types. It can greatly amortize the design and development costs of the reputation system.

5. Semi-distributed. This system features a central server that maintains user accounts and answers reputation inquiries. All application-related QoS information, however, is stored across system users either in a random fashion or through a distributed hash table (DHT).

6. Secure. This system can protect the privacy of references and withstand various misbehavior, such as defamation and flattery, using lightweight

techniques like multivariate outlier detection [50] and symmetric-key cryptographic functions.

## 1.3 CASE STUDY OF REPUTATION SYSTEMS

### 1.3.1 PowerTrust

PowerTrust [51] is a robust and scalable P2P reputation system that uses a *trust overlay network* (TON) to model the trust relationships among peers. The authors of PowerTrust, Zhou and Hwang, first examined eBay transaction data from over 10,000 users, and discovered a power-law distribution in user feedback. Their mathematical analysis justifies that a power-law distribution effectively models any dynamically growing P2P feedback systems, whether structured or unstructured. The authors then developed the PowerTrust system to leverage the power-law feedback characteristics of P2P systems. The PowerTrust system dynamically selects a small number of the most reputable nodes as determined by a distributed ranking mechanism; these nodes are termed *power nodes*. Using a look-ahead random walk strategy and leveraging power nodes, PowerTrust significantly improves on previous systems with respect to global reputation accuracy and aggregation speed. PowerTrust is adaptable to highly dynamic networks and robust to disturbances by malicious peers.

As one of the major building blocks in a PowerTrust system, a TON is built on top of all peers in a P2P system. It is a virtual network represented by a directed graph on top of a P2P system. The graph nodes correspond to the peers. The directed edges or links are labeled with the feedback scores between two interacting peers. Whenever a transaction takes place between a peer pair, each peer in the pair evaluates the other. Therefore, all peers frequently send *local trust scores* among themselves. These scores are considered the raw data input to the PowerTrust system. The system aggregates the local scores to calculate the global reputation score of each participating peer. All global scores form a *reputation vector*, $V = (v_1, v_2, v_3, \ldots, v_n)$, which is the output of the PowerTrust system. All global scores are normalized with $\sum_i v_i = 1$, where $i = 1, 2, \ldots, n$ and $n$ is the TON network size.

The system is built with five functional modules. The *regular random walk* module supports the *initial reputation aggregation*. The *look-ahead random walk* (LRW) module is used to periodically *update reputation scores*. To this end, the LRW also works with a *distributed ranking module* to identify power nodes, which are leveraged to update global reputation scores.

In PowerTrust, feedback scores are generated by Bayesian learning [47] or by an average rating based on peer satisfaction. Each node normalizes all issued feedback scores and stores them in a *trust matrix*. Consider the trust matrix $R = (r_{ij})$ defined over an $n$-node TON, where $r_{ij}$ is the *normalized local trust score* defined by $r_{ij} = s_{ij} / \sum_j s_{ij}$ and $s_{ij}$ is the most recent feedback score about node $j$ from node $i$. If there is no link from node $i$ to node $j$, $s_{ij}$ is set to 0. Therefore, for all $1 \leq i, j \leq n,$, $0 \leq r_{ij} \leq 1$, and $\forall i \sum_{j=1}^{n} r_{ij} = 1$. In other words, matrix $R$ is a stochastic matrix, in which all entries are fractions and each row sum equals 1. This demands that the scores issued by the same node to other peers are normalized.

All global reputation scores $v_i$ for $n$ nodes form a *normalized reputation column vector* $V = (v_i)$, where $\sum_i v_i = 1$. The reputation vector V is computed by Equation (1.1) and given an arbitrary initial reputation vector $V_{(0)}$ and small error threshold $\varepsilon$. For a system of $n$ nodes, the authors initialized $v_i = 1/n$. For all $t = 1, 2, \ldots, k$, while $|V_{(i)} - V_{(i-1)}| > \varepsilon$, the successive reputation vectors are computed recursively by:

$$V_{(t+1)} = R^T \times V_{(t)} \tag{1.1}$$

After a sufficient number $k$ iterations, the global reputation vector converges to the eigenvector of the trust matrix $R$ [11].

The LRW strategy efficiently aggregates global reputations. Each node in the TON not only holds its own local trust scores but also aggregates its neighbors' first-hand trust scores. The enhanced trust matrix $S$ is computed by $S = R^2$. The extra aggregation overhead grows linearly in sparse power-law graphs [52]. PowerTrust uses a DHT to implement its distributed ranking mechanism. Every node has a score manager that accumulates its global reputation. When a new node $i$ joins the system, the successor node of $k_i$ is assigned to be the score manager of node $i$, where $k_i$ is the hash of node $i$'s unique identifier by a pre-defined hash function. All other nodes can access the global reputation of node $i$ by issuing a lookup request with the key equal to $k_i$. Several different hash functions can be used to enable multiple score managers for each node in case a malicious score manager reports false global reputation scores.

Each node $i$ sends all local trust scores to the score managers of its out-degree neighbors for the initial global reputation aggregation. After the first round of aggregation, the score managers collaborate with each other to find power nodes. Triplet $(i, v_i, j)$ means that node $i$ has global reputation score $v_i$ and node $j$ is node $i$'s score manager. If node $x$ stores the triplet $(i, v_i, j)$ and finds $i$ to be a power node, node $x$ will notify node $i$'s score manager, $j$. Because the trust matrix $R$ is dynamically changing as new peers join and new transactions occur, the global reputation scores should be updated periodically, especially for power nodes. These global reputation updates leverage the capabilities of power nodes.

PowerTrust is distinguished by its fast reputation aggregation, ranking, updating, system scalability and wide applicability, and system robustness and operational efficiency. Experimental results have confirmed PowerTrust's effectiveness and robustness for use in a wide variety of P2P applications.

### 1.3.2 SocialTrust

SocialTrust [53] leverages social networks to enhance the effectiveness of current reputation systems in combating collusion. A social network is a social structure consisting of individuals (nodes) that are linked by one or more specific types of relationships, such as common interests, friendship, or kinship [54].

To investigate the impact of a social network on user purchasing and rating patterns, the authors of SocialTrust analyzed a real trace of 450,000 transaction ratings during 2008 - 2010 crawled from Overstock Auctions (abbreviated to Overstock). Overstock is an online auction platform similar to eBay, but it distinguishes itself by integrating a social network into the market community. The authors found that *social closeness* and *interest similarity* impact user purchasing and rating patterns. First, users tend to buy products from high-reputed users. Second, users tend to buy products from socially-close (3 hops or less) users, and give socially-close users high ratings. Third, 88% of a user's purchases are within 20% of the user's product interest categories on average, and 60% of transactions are conducted between users sharing >30% interest similarity. Based on these observations, suspicious collusion behavior patterns were identified from the distance and interest relationships between peers in a social network:

1. (B1) Socially distant users frequently and highly rate each other.
2. (B2) Users frequently and highly rate low-reputed socially-close users.
3. (B3) Users with few common interests frequently and highly rate each other.
4. (B4) Users frequently give common-interest users low ratings.

SocialTrust derives *social closeness* (denoted by $\Omega_d$) from the social network graph and node interactions and *interest similarity* (denoted by $\Omega_c$) from node profiles or activities between a pair of nodes. SocialTrust detects action patterns of suspicious collusion behaviors and then reduces the weight of the ratings from suspected colluders based on the two coefficients.

The authors first introduced a method to calculate the social closeness between two adjacent nodes in a social network, and then introduced a method for non-adjacent nodes having no direct social relationship. The closeness of a pair of nodes $n_i$ and $n_j$ is determined by two factors: the number of social relationships and the interaction frequency. Therefore, the social closeness $\Omega_{d_{(i,j)}}$ between two adjacent nodes $n_i$ and $n_j$ is calculated by

$$\Omega_{d_{(i,j)}} = \frac{m_{(i,j)} f_{(i,j)}}{\sum_{k=0}^{|\mathscr{S}_i|} f_{(i,k)}}, \tag{1.2}$$

where $m_{i,j} \geq 1$ denotes the number of social relationships between $n_i$ and $n_j$, $f_{i,j}$ denotes the interaction frequency from $n_i$ to $n_j$, $\mathscr{S}_i$ denotes a set of neighbors of node $i$, and $|\mathscr{S}_i|$ denotes the number of neighbors in the set of $\mathscr{S}_i$. Using $\mathscr{S}_i$ and $\mathscr{S}_j$ to denote the set of friends of two non-adjacent nodes $n_i$ and $n_j$, respectively, the social closeness between $n_i$ and $n_j$ is defined as:

$$\Omega_{d_{(i,j)}} = \sum_{k \in |S_i \cap S_j|} \frac{\Omega_{d_{(i,k)}} + \Omega_{d_{(k,j)}}}{2} \tag{1.3}$$

That is, the authors found all the common friends $n_k$ between $n_i$ and $n_j$. The social closeness between $n_i$ and $n_j$ through $n_k$ is calculated by averaging the closeness of $\Omega_{(i,k)}$ and $\Omega_{(k,j)}$.

According to *B3* and *B4*, when $n_i$ gives $n_j$ high ratings with high frequency, if $\Omega_{d_{(i,j)}}$ is very low or very high and $n_j$'s reputation is low, $n_i$ is potentially a colluder. Then, SocialTrust reduces the weight of the ratings from $n_i$ to $n_j$ based on $\Omega_{d_{i,j}}$.

As shown in Figure 1.1, SocialTrust uses the Gaussian function to adjust the ratings from $n_i$ to $n_j$, denoted by $r_{(i,j)}$.

$$r_{(i,j)} = r_{(i,j)} \cdot \alpha \cdot e^{-\frac{(\Omega_{d_{(i,j)}} - \bar{\Omega}_{d_i})^2}{2|\max \Omega_{d_i} - \min \Omega_{d_i}|^2}}, \tag{1.4}$$

where $\alpha$ is the function parameter and $\max \Omega_{d_i}$, $\min \Omega_{d_i}$ and $\bar{\Omega}_{d_i}$ denote the maximum, minimum and average social closenesses of $n_i$ to other nodes that $n_i$ has rated.

**Figure 1.1** One-dimensional reputation adjustment.

The exponent in Equation (1.4) is the deviation of the social closeness of $n_i$ and $n_j$ from the normal social closeness of $n_i$ to other nodes it has rated. As Figure 1.1 shows, the Gaussian function significantly reduces the weights of the ratings from nodes with very high or very low social closeness to the rated nodes and mildly reduces the weights of those from the nodes with high or low social closeness to the rated nodes, while nearly maintaining the ratings from the nodes with normal closeness to the rated nodes. As a result, the weights of the ratings from suspected colluders are reduced.

In SocialTrust, each node has an interest vector $\mathscr{V}{=}{<}v_1,v_2,v_3,...,v_k{>}$ indicating its interests. The social interest similarity of $n_i$ to $n_j$ is calculated by:

$$\Omega_{c(i,j)} = \frac{|\mathscr{V}_i \cap \mathscr{V}_j|}{\min(|\mathscr{V}_i|,|\mathscr{V}_j|)}. \tag{1.5}$$

According to *B3* and *B4*, SocialTrust reduces the weights of the ratings from suspected colluders that have very high or low $\Omega_{c(i,j)}$ with the rated node using the Gaussian function:

$$r_{(i,j)} = r_{(i,j)} \cdot \alpha \cdot e^{-\frac{(\Omega_{c_{(i,j)}} - \bar{\Omega}_{c_i})^2}{2|\max \Omega_{c_i} - \min \Omega_{c_i}|^2}}, \tag{1.6}$$

where $\max \Omega_{c_i}$, $\min \Omega_{c_i}$ and $\bar{\Omega}_{c_i}$ denote the maximum, minimum and average interest similarity of node $n_i$ with the nodes it has rated, respectively. The rating from $n_i$ to $n_j$ is adjusted according to Equation (1.6) when $n_i$ frequently rates $n_j$ with high ratings and $(\Omega_{c_{(i,j)}} - \bar{\Omega}_{c_i})^2 < 0$, which implies that $n_i$ and

$n_j$ share few interests, or when $n_i$ frequently rates $n_j$ with low ratings and $(\Omega_{c_{(i,j)}} - \bar{\Omega}_{c_i})^2 > 0$, which implies that $n_i$ and $n_j$ share many interests. Combining Formulas (1.4) and (1.6), with simultaneous consideration of social closeness and interest similarity, the reputation can be adjusted by:

$$r_{(i,j)}(\Omega_d, \Omega_c) = r_{(i,j)} \cdot \alpha \cdot e^{-\left(\frac{(\Omega_{d_{(i,j)}} - \bar{\Omega}_{d_i})^2}{2|\max \Omega_{d_i} - \min \Omega_{d_i}|^2} + \frac{(\Omega_{c_{(i,j)}} - \bar{\Omega}_{c_i})^2}{2|\max \Omega_{c_i} - \min \Omega_{c_i}|^2}\right)}. \tag{1.7}$$

Let $H_d$ and $L_d$ denote very high and low social closeness, and let $H_c$ and $L_c$ denote very high and low interest similarity. Then, the rating values between the nodes that have $(H_d, H_c)$, $(H_d, L_c)$, $(L_d, H_c)$ and $(L_d, L_c)$ are greatly reduced. Therefore, based on Formula (1.7), the influences of the collusion listed in *B1*-*B4* are reduced.

In reputation systems, each resource manager is responsible for collecting the ratings and calculating the global reputation of certain nodes. Thus, each resource manager can keep track of the rating frequencies and values of the nodes it manages, which helps them to detect collusion in SocialTrust. A manager adjusts the ratings from suspected colluders when calculating global node reputation periodically.

Experiment results show SocialTrust greatly enhances the capability of eBay's reputation system and EigenTrust in countering collusion. SocialTrust can even detect colluders when compromised pre-trusted and high-reputed nodes are involved in collusion.

### 1.3.3 Accurate Trust Reflection

Shen and Zhao [55] claimed that existing reputation systems fall short in accurately reflecting node trust and providing the right guidance for server selection for two main reasons. First, they directly regard node reputation as trust, which is not appropriate in general. Reputation represents the opinion formed by other nodes about a node's QoS behavior, while trust is an assessment of a node's honesty and willingness to be cooperative. Because the additional factors other than trust (e.g., capacity and longevity) that contribute to reputation are heterogeneous and time-varying attributes, a node's reputation does not reflect its trust and its current QoS. Benign but overloaded nodes may get low reputations due to insufficient capacity or overwhelming service requests. Benign but low-longevity nodes may also have low reputation values due to their short longevity. Second, they guide a node to select the server with the highest reputation, which may not actually select a high-QoS server and would overload the highest-reputed nodes.

**Figure 1.2** The proposed trust system.

The authors used an experiment to study the relationship reputation has with capacity and longevity. In this experiment, all nodes are trustworthy with a 100% probability of serving requests, and they each receive approximately the same number of requests. A node's available capacity equals $c_a = c - w$, where $c$ is its capacity represented by the number of service requests it can handle during a time unit, and $w$ is its workload represented by the number of its received requests during a time unit. Based on the observation, the authors developed an optimal server selection algorithm that separately considers node trust and the current values of additional factors to ensure high QoS (Figure 1.2).

**Figure 1.3** Reputation vs. capacity. **Figure 1.4** Reputation vs. longevity.

Firstly, the authors tested the relationship between node reputation and capacity with the assumption that each server's longevity is high enough to complete the requested services. In this case, since a higher available capacity enables a server to offer higher QoS, a node gives its server a reputation value of $r_c$, which equals the server's available capacity $c_a$. Figure 1.3 shows the reputation of each node versus its capacity. It demonstrates that node capacity positively influences a node's reputation.

Secondly, the authors tested the relationship between node reputation and node longevity with the assumption that each server's available capacity is high enough to complete the services requested from it. In this case, a server's reputation should be evaluated according to the percent of the requested services it has completed. Thus, the reputation value $r_l$ equals

$$r_l = \begin{cases} l_a/l_r & \text{if } l_a/l_r < 1 \\ 1 & \text{otherwise,} \end{cases} \tag{1.8}$$

where $l_a$ is the available longevity of a server and $l_r$ is the time requirement of a requested service. Figure 1.4 shows the reputation of each node versus its longevity. It demonstrates that the relationship between reputation and longevity exhibits a logarithmic trend, and higher-longevity nodes have higher reputations.

The experiment results confirm that a reputation value itself cannot directly reflect node trust, since trust is also affected by capacity and longevity. Thus, reputation cannot be directly used for optimal server selection. The effect of a node's previous capacity and longevity on the reputation should be removed when evaluating a node's trustworthiness, and the currently available capacity and longevity should be considered with trust in choosing a service provider.

The proposed trust models can be built on any reputation system. The resource managers build the trust models, which need information on additional factors when evaluating node trust. Thus, each node $i$ periodically sends its resource manager a vector including its current values for the additional factors that affect reputation, denoted $V_i =< c_a, l_a, \cdots >$ using `Insert(`$ID_i$`,`$V_i$`)`, where $c_a$ and $l_a$ are the node's current available capacity and available longevity, respectively. After the resource manager computes the global reputation value of node $i$, it uses $V_i$ to derive its trustworthiness using the proposed trust models.

The authors introduced two trust models: manual and automatic. These models help to remove the influence of node capacity and longevity on reputation when determining node trust.

Since the reputation has a linear relationship with capacity, if there are no other factors that influence reputation except capacity, the ratio of a node's reputation over its capacity can be used to measure its trust, i.e., $t_c = r/c$, where $r$ and $c$ denote the node's reputation and capacity, respectively. $t_c$ represents the reputation earned by a node for each unit of capacity it has contributed to providing service. The authors assumed there are $m$ levels of trust in the system. The normalized $t_c$ determines a node's trust level. Figure 1.5 shows a coordinate graph with the $x$ axis representing capacity and the $y$ axis representing reputation. The space is divided into different sections, each representing a trust level. A higher trust level means a node's reputation is high relative to its capacity. The authors mapped a node's normalized $t_c$

to the graph according to its capacity. Based on its coordinate location, the node's trust level is determined. Figure 1.6 shows that the reputation has a logarithmic relationship with longevity. Using MATLAB, the logarithmic curve is transformed to a line by changing longevity $l$ to $loglog(l+1)$. Hence, reputation has a linear relationship to $loglog(l+1)$. As with capacity, the authors used $r/loglog(l+1)$ to measure a node's trust level when there are no other additional factors except longevity. By considering both capacity and longevity, the authors introduced spatial/temporal values. By viewing node capacity in a spatial domain and node longevity in a temporal domain, the spatial/temporal value $u$ is defined as:

$$u = \alpha \times (c \cdot loglog(l+1)),$$

where $\alpha$ is a constant factor. Based on the above analysis, the relationship between reputation $r$ and $u$ can be approximately regarded as a linear relationship. A node with a higher $u$ should have a higher reputation and vice versa. Thus, each resource manager builds a manual trust model as shown in Figure 1.6. The model is a two-dimensional space where spatial/temporal value and reputation are coordinates. A node's trust value is calculated by $t_v = r/u$. Locality-preserving hashing [56] is used to normalize the trust value in order to obtain trust level $t_l$. That is, $t_l = m \cdot t_v/(\max(t_v) - \min(t_v))$, where $m$ is the number of trust levels in the system, and $\max(t_v)$ and $\min(t_v)$ are the maximum and minimum values of $t_v$ in the system, respectively. The authors mapped a node's normalized $t_l$ to the graph according to its $u$. Based on its coordinate location, the node's trust level is determined.

**Figure 1.5** Capacity-based trust.

**Figure 1.6** Capacity & longevity-based trust.

The automatic trust model uses a neural network technique [57] for node trust evaluation by catching the nonlinear relationship between reputation, trust, and additional factors including capacity and longevity. The authors built a neural network with one layer of hidden units [58] as shown in Figure 1.7.

**Figure 1.7** Neural network based trust model.

It is a nonlinear function from a set of input variables $P = \{p_1, p_2, \cdots\}$ to output variable $y \in [0,10]$ controlled by a set of vectors of adjustable weight parameters $W(w_{ji} : 1 \leq i \leq n, 1 \leq j \leq k)$. The input units are a node's attribute variables including reputation, longevity, capacity and other additional factors, and the output is the node's trust level. The activation of a hidden unit is a function $F_i$ of the weighted inputs plus a bias as given in the following equation:

$$y(P,W) = F_j(\sum_{j=1}^{k} w_{1j}^{(2)} F_i(\sum_{i=1}^{n} w_{ji}^{(1)} p_i + b_1) + b_2),$$

where

$$F(x) = \frac{1}{1+e^{-x}}.$$

After training, a resource manager can directly use the trained neural network for trust evaluation of its responsible nodes.

The proposed optimal server selection algorithm considers the trust derived by the trust model and the current available capacity and longevity. When choosing a server, a client first identifies all servers with sufficient capacity and longevity to meet its needs. It then chooses the nodes from these options that have the highest trust. These selected servers can reliably satisfy the client's request. Then, in order to distribute the load among nodes according to their available capacity without overloading a server, lottery scheduling [59] is adopted in the final server selection so that servers with higher available capacity receive more requests and vice versa.

## 1.4 OPEN PROBLEMS

Though significant research has been conducted in reputation management systems, there still exists a number of open problems in this area that need to be resolved.

The first problem is handling intelligent adversaries. An intelligent adversary can manipulate the policies of the reputation system, adjusting the degree of his misbehavior such that he can evade the misbehavior detection system. For example, several nodes may collude to boost their reputation values. Malicious nodes control a large pool of sub-nodes in order to dominate a large portion of the identity space so as to increase their power in voting, leaving other nodes vulnerable to denial of service attacks. Although a number of research works have been conducted to detect collusion or Sybil attacks based on their interaction graphs or social relationships, this interaction and social information can still be manipulated by intelligent adversaries.

The second problem is estimating node reputation quickly. In most reputation systems, it takes a while to build reputation and trust among nodes in a network. Minimizing this startup period is still an open issue. Most reputation systems investigate a system at a stable stage where the reputation value of nodes are already calculated based on their historical interaction behaviors. Since it takes a long time to accumulate reputation values of nodes in order to identify malicious nodes, during this time, a malicious node may cause serious damage to the system. Therefore, ways to identify malicious nodes in a short time, predict the trustworthiness of a newly joined node, and design an effective access control mechanism are also very important.

The third problem is truly reflecting node trustworthiness in reputation systems. Most reputation systems calculate the reputation values of peers based on their previous behaviors. A good behavior leads to an increased reputation while a bad behavior leads to a decreased reputation. However, sometimes the misbehavior of users is not the results of malicious intentions but rather system errors such as network congestion or I/O errors. A method to distinguish between causes of misbehavior is an interesting problem to be investigated in the future.

The fourth problem is reputation evaluation in a social network environment. Considering social trustworthiness between users, the ratings of common-interest nodes or friends should be considered more trustable. Qualitatively estimating the influence of social closeness on the trust between users and on server selection is also an interesting problem.

## 1.5 CONCLUSION

P2P networks play an important role nowadays, especially for data sharing and digital media distribution like video on demand. In order to enhance the robustness, availability, and security of P2P networks, reputation management systems are used as a tool to detect malicious peers and provide cooperation incentives in P2P networks. Numerous technologies based on reputation systems have been proposed. These advances were designed to solve practical and challenging problems, including collusion, content pollution, free-riding and P2P network attacks.

In this chapter, we have classified the main works on reputation systems into two categories, scalability/accuracy and security, and presented a comprehensive review of research works in each category. In the case studies, we have presented three representative reputation systems that enable high scalability, collusion avoidance, and accurate trust reflection. Finally, we have briefly discussed open problems in the research area of reputation systems. Challenges in open problems as well as the difficulty in implementing of complex reputation systems introduce many problems in P2P networks. Solutions to these problems will accelerate the maturation of P2P networks, creating a new generation of secure, trustworthy P2P applications.

# ACKNOWLEDGMENTS

This research was supported in part by U.S. NSF grants NSF-CSR 1025649, OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

## REFERENCES

1. KaZaA. http://www.kazaa.com/.

2. BitTorrent. http://www.bittorrent.com/.

3. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Frans Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 2003.
4. A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of Middleware*, 2001.
5. Skype. http://www.skype.com/.
6. X. Wang, Z. Yao, Y. Zhang, and D. Loguinov. Enhancing Application-Layer Multicast for P2P Conferencing. In *Proc. of IEEE Consumer Communications and Networking Conference*, 2007.
7. H. Shen, Z. Li, T. Li, and Y. Zhu. PIRD: P2P-Based Intelligent Resource Discovery in Internet-Based Distributed Systems. In *Proc. of International Conference on Distributed Computing Systems*, 2008.
8. eBay. http://www.ebay.com/.
9. Amazon. http://www.amazon.com/.
10. Overstock. http://www.overstock.com/.
11. S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proc. of The 12th International World Wide Web Conference*, 2003.
12. Q. Lian, Z. Zhang, M. Yang, B. Zhao, Y. Dai, and X. Li. An Empirical Study of Collusion Behavior in the Maze P2P File-Sharing System. In *Proc. of International Conference on Distributed Computing Systems*, 2007.
13. S. Zhao and V. Lo. Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems. In *Proc. of Peer-to-Peer Computing*, 2005.

14. L. Xiong and L. Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 2004.

15. M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proc. of International World Wide Web Conference*, 2005.

16. S. Song, K. Hwang, R. Zhou, and Y. Kwok. Trusted P2P Transactions with Fuzzy Reputation Aggregation. *IEEE Internet Computing*, 2005.

17. K. Walsh and E. Sirer. Fighting Peer-to-Peer SPAM and Decoys with Object Reputation. In *Proc. of Economics of P2P Systems*, 2005.

18. C. Costa, V. Soares, J. Almeida, and V. Almeida. Fighting Pollution Dissemination in Peer-to-Peer Networks. In *Proc. of Symposium On Applied Computing*, 2005.

19. C. Costa and J. Almeida. Reputation Systems for Fighting Pollution in Peer-to-Peer File Sharing Systems. In *Proc. of Peer-to-Peer Computing*, 2007.

20. J. Douceur. The sybil attack. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems*, 2002.

21. R. Zhou, K. Hwang, and M. Cai. Gossip-Based Reputation Management for Unstructured Peer-to-Peer Networks. *IEEE Transactions on Knowledge and Data Engineering*, 2007.

22. D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *Proc. of IEEE Symposium on Foundations of Computer Science*, 2003.

23. R. Zhou, K. Hwang, and M. Cai. Gossip Trust for Fast Reputation Aggregation in Peer-to-Peer Networks. *IEEE Transactions on Knowledge and Data Engineering*, 2008.

24. Z. Liang and W. Shi. PET: A PErsonalized Trust Model with Reputation and Risk Evaluation for P2P Resource Sharing. In *Proc. of Hawaii International Conference on System Sciences*, 2005.

25. H. Zhao and X. Li. H-Trust: A Robust and Lightweight Group Reputation System for Peer-to-Peer Desktop Grid. In *Proc. of International Conference on Distributed Computing Systems Workshops*, 2008.

26. J. Hirsch. An Index to Quantify an Individual's Scientific Research Output. In *Proc. of the National Academy of Sciences*, 2005.

27. S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative Peer Groups in NICE. In *Proc. of IEEE International Conference on Computer Communications*, 2003.

28. A. Selcuk, E. Uzun, and M. Pariente. A Reputation-based Trust Management System for P2P Networks. In *Proc. of IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2004.

29. A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proc. of Peer-to-Peer Computing*, 2003.

30. M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proc. of ACM Conference on Electronic Commerce*, 2004.

31. T. Moreton and A. Twigg. Trading in Trust, Tokens, and Stamps. In *Proc. of The 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

32. D. Fudenberg and D. Levine. The Theory of Learning in Games. *The MIT Press*, 1999.

33. E. Damiani, S. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *Proc. of the 9th ACM Conference on Computer and Communications Security*, 2002.

34. N. Curtis, R. Safavi-Naini, and W. Susilo. $X^2$Rep: Enhanced Trust Semantics for the XRep Protocol. In *Proc. of International Conference on Applied Cryptography and Network Security*, 2004.

35. E. Zhai, R. Chen, Z. Cai, L. Zhang, E. Lua, H. Sun, S. Qing, L. Tang, and Z. Chen. Sorcery: Could We Make P2P Content Sharing Systems Robust to Deceivers? In *Proc. of Peer-to-Peer Computing*, 2009.

36. F. Cornelli, E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati. Choosing Reputable Servents in a P2P Network. In *Proc. of the 11th World Wide Web Conference*, 2002.

37. Q. Zhang, W. Wei, and T. Yu. On the Modeling of Honest Players in Reputation Systems. *Computer Science and Technology*, 2009.

38. Y. Zhang, S. Chen, and G. Yang. Sftrust: A Double Trust Metric Based Trust Model in Unstructured P2P System. In *Proc. of IEEE International Parallel and Distributed Processing Symposium*, 2009.

39. Y. Sun, Z. Han, and K.J.R. Liu. Defense of Trust Management Vulnerabilities in Distributed Networks. *Communications Magazine, IEEE*, 2008.

40. H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. Sybilguard: Defending against Sybil Attacks via Social Networks. In *Proc. of ACM Special Interest Group on Data Communication*, 2006.

41. L. Lu, J. Han, Y. Liu, L. Hu, J. Huai, L. Ni, and J. Ma. Pseudo Trust: Zero-Knowledge Authentication in Anonymous P2Ps. *IEEE Transactions on Parallel and Distributed Systems*, 2008.

42. V. Scarlata, B. Neil Levine, and C. Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. In *Proc. of IEEE International Conference on Network Protocols*, 2001.

43. Q. Feng, Y. Yang, Y. Sun, and Y. Dai. Modeling Attack Behaviors in Rating Systems. In *Proc. of International Conference on Distributed Computing Systems Workshops*, 2008.

44. Y. Sun and Y. Yang. Trust Establishment in Distributed Networks: Analysis and modeling. In *Proc. of IEEE International Conference on Communications*, 2007.

45. T. Papaioannou and G. Stamoulis. Achieving Honest Ratings with Reputation-Based Fines in Electronic Markets. In *Proc. of IEEE International Conference on Computer Communications*, 2008.

46. P. Dewan and P. Dasgupta. P2P Reputation Management Using Distributed Identities and Decentralized Recommendation Chains. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

47. S. Buchegger and J. Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proc. of Economics of Peer-to-Peer Systems*, 2004.

48. Y. Zhang and Y. Fang. A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 2007.

49. A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 1995.

50. S. Bay and M. Schwabacher. Mining Distance-Based Outliers in Near Linear Time with Randomization and A Simple Pruning Rule. In *Proc. of International Conference Knowledge Discovery and Data Mining*, 2003.

51. R. Zhou and K. Hwang. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems*, 2007.

52. M. Mihail, A. Saberi, and P.Tetali. Random Walks with Lookahead in Power Law Random Graphs. In *Proc. of International World Wide Web Conference*, 2004.

53. Z. Li, H. Shen, and K. Sapra. Leveraging Social Networks to Combat Collusion in Reputation Systems for Peer-to-Peer Networks. In *Proc. of IEEE International Parallel and Distributed Processing Symposium*, 2011.

54. M. Mcpherson. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 2001.

55. H. Shen and L. Zhao. Refining Reputation to Truly Select High-QoS Servers in Peer-to-Peer Networks. In *Proc. of the IEEE 20th International Conference on Computer Communications and Networks*, 2011.

56. M. Cai, M. Frank, and P. Szekely. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. *Journal of Grid Computing*, 2004.

57. C. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

58. K. Hornik, M. Stinchcombe, and H. White. Multilayer Feed-forward Networks are Universal Approximators. *Neural Networks*, 1989.

59. C. Waldspurger and W. Weihl. Lottery Scheduling: Flexible Proportional-Share Resource Management. In *Proc. of USENIX Symposium on Operating Systems Design and Implementation*, 1994.