Trust by Association:
A Meta-Reputation System for Peer-to-Peer Networks

Matthew Kellett
Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario
K1A 0Z4 Canada
W: (613) 991-4362
F: (613) 993-9940

Thomas Tran
School of Information Technology and Engineering
University of Ottawa
800 King Edward Avenue
Ottawa, Ontario
K1N 6N5 Canada
W: (613) 562-5800 ext. 6215
F: (613) 562-5175

Ming Li
Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario
K1A 0Z4 Canada
W: (613) 991-1388
F: (613) 993-9940

November 2008
Revised: April 2009
Revised: November 2009
Final:April 2010

Abstract

Trust mechanisms are used in peer-to-peer (P2P) networks to help well-behaving peers find other well-behaving peers with which to trade. Unfortunately, these trust mechanisms often do little to keep badly-behaving peers from entering and taking advantage of the network, which makes the resulting network difficult or impossible to use for legitimate purposes such as e-commerce. We propose *trust by association*, a way of tying peers together in invitation-only P2P networks in such a way as to encourage the removal of badly-behaving peers. We use invitations to create a structure within the otherwise ad hoc P2P network. Using this structure, we create a *meta-reputation* system where we measure a peer's trustworthiness not only by its own behaviour, but also by the behaviour of the peers it has invited to join. The connection created between the peers takes advantage of the external social relationship that must exist before a peer can be invited into the network. The result is a P2P network where, rather than just trying to marginalize badly-behaving peers, there is incentive to kick them out of the network. We present results from a simple simulation showing that our approach works well in general when combined with and compared to an existing trust mechanism.

KEYWORDS: trust, peer-to-peer networks, reputation, invitation-only, e-commerce, malicious peers

1 Introduction

Trust mechanisms in peer-to-peer (P2P) networks attempt to reduce the impact of malicious and selfish peers by rewarding honest peers that participate fully in the network. A malicious peer is one that provides corrupt or infected resources, poor quality services, and/or dishonest opinions about the trustworthiness of other peers. A selfish peer is one that tries to benefit from the network without providing any resources to it. Trust mechanisms attempt to quantify the behaviour of a peer in order to give an assessment of its trustworthiness, which we define as the likelihood that the next transaction with that peer will be successful. A peer is chosen for a transaction with probability proportional to its trustworthiness from amongst peers offering the same resource or service, which means that a well-designed trust mechanism will reduce the number of interactions with malicious and selfish peers in the network.

We propose a way to provide incentive not only to reduce interactions with malicious and selfish peers but to exclude such peers entirely from the network. We accomplish this goal by taking advantage of the social network that must inevitably accompany an invitation-only P2P network. Invitation-only networks create an environment where we can measure a peer's trustworthiness not just by its own behaviour but also by the behaviour of the peers it has invited to join the network. We show how combining the trust values, generated using an existing algorithm, of a peer and its invitees gives the peer incentive to disassociate itself from dishonest peers. We want to combine the results of existing reputation-based trust algorithms into a meta-reputation system rather than replace them, so that the network benefits from both the properties of the existing algorithm and the incentives provided by our approach.

We believe that invitation-only networks provide an excellent platform from which to do e-commerce in P2P networks. An attractive e-commerce application for this type of network is one that allows for the legal trading of copyrighted material such as music or movies and which is sanctioned by industry groups such as the Recording Industry Association of America (RIAA) or the Motion Picture Association of America (MPAA). Using a micro-payment scheme, each transaction would cost from a few cents upward with the cost being tied to the trustworthiness of the peer, giving peers even more incentive to act honestly. In return, the participants in the network get a guilt-, harassment-, and lawsuit-free environment while the industry groups get a piece of the P2P trading pie.

We start by looking in detail at the existing approaches to trust and their properties in P2P networks in Section 2. We then present a wide-ranging discussion of our approach and variations on it in Section 3. In Section 4, we analyze our equations to make sure that they react as intended and present the results of a simple simulation comparing trust by association to EigenTrust. Finally, we conclude with a discussion of future work in Section 5.

2 Background and related work

2.1 Existing P2P networks

Trust mechanisms in P2P networks attempt to reduce the interactions with malicious and selfish peers. In order to understand how trust mechanisms accomplish this reduction, we must first understand how these networks function in general. A P2P network is a collection of computer users on the same computer network with a common interest who join together to pool their resources. By far the most popular application for P2P networks is the sharing of files, especially music, movies, and TV shows. It is easy to see the benefit of this type of system. Alone, a single computer user may have the chance to record a handful of TV shows a week and may miss some weeks because she is busy or forgetful. Together with hundreds of other users in a P2P network, there is a good chance that other users are interested in the same show and have recorded the show that the user missed. Each peer makes a small effort and all peers reap the rewards.

Not all peers act altruistically. There are some peers that seek to take advantage of the unorganized, ad hoc nature of P2P networks. Some of these users actively try to disrupt the sharing. There are P2P networks that have been organized specifically to trade material that is almost always copyrighted, such as the music, movies, and TV shows we mentioned above. Trading in P2P networks does not fall under

the fair-use clauses of most countries' copyright legislation. As an example, someone sharing music on a P2P network is obviously engaged in a different activity from someone making a mixed tape of music for her friends. There has thus been considerable effort on the part of the industry associations that represent the copyright holders to try to thwart these networks. The industry associations' stated goal is to join the networks—since anyone can join—and flood them with corrupt and useless versions of the material the associations are trying to protect. The idea is that the user looking for a TV show will only download so many bad copies of it before giving up. Still other users are interested in P2P networks as distribution networks for viruses. They distribute valid but infected copies of these files in the hope of infecting more computers. Finally, there are users that try to benefit from the network without sharing resources. It is easy to see that a user needs to offer something to the network in order for the network to be viable. If no users offered anything, there would be no reason for the network to exist.

These malicious and selfish users can exist in the network because of how it works. When our user goes looking for her TV show, she sends a query through the network. Depending on the way the network is organized, the query will either travel through the entire network or to a subset of the peers. All the peers that have the requested resource respond to the query. The user then picks from amongst the peers that have responded, and downloads the TV show. Alternatively, the user's peer software may randomly choose and download the TV show for her. The result either way is essentially a random choice. There is no way of knowing whether the peer is trustworthy or not. In the case of a TV show or movie, the downloading of the file can take hours. If at the end of the download the file turns out to be corrupt or the user's anti-virus software (if she is lucky) flags the file as infected, the user has wasted a lot of time for no gain. Even worse, files that are downloaded are generally automatically offered for upload, so if the user does not check the file, she may unknowingly propagate the corrupt or infected file.

2.2 Reputation-based trust mechanisms

Trust mechanisms work by assigning a trust value to each peer based on the peer's history of transactions. We will refer to the peer looking for the resource as the *querying peer* and the peer or peers that respond to the query as the *offering peer* or *offering peers*. We are interested in the trust values of the offering peers. We will focus on reputation-based trust mechanisms, where a querying peer asks other peers for their opinion of the trustworthiness of an offering peer, as opposed to a experience-based trust mechanism, where the querying peer relies solely on previous transactions it has had with the offering peer. Instead of randomly picking from amongst the offering peers, the user can now use their trust values to probabilistically choose an offering peer. The higher the peer's trust value, the more likely that it will be picked for the transaction. This approach lowers the chances that a malicious or selfish peer will be chosen for the transaction and has the effect of marginalizing such peers.

Probabilistically choosing an offering peer would seem to be risky since there is some possibility that a malicious peer will be chosen. It seems simpler to always choose the most trustworthy offering peer. Always choosing the most trustworthy peer is good strategy for an individual peer but bad strategy for the network as a whole. Amongst the problems with this strategy is that it causes a de facto denial of service attack against the most trustworthy offering peer. Most trust mechanisms assign trust on the basis of the volume of transactions, meaning that a peer that has successfully completed 20 transactions is considered more trustworthy than a peer that has completed 2. If the most trustworthy peer is always chosen that peer will become the only source on the network for all the resources it holds regardless of the fact that similarly honest peers hold the same resources elsewhere in the network. Probabilistically choosing the peer evens out this effect and distributes the transactions across honest peers.

Why not simply eliminate the peers with low trust values from consideration entirely, since they are obviously malicious or selfish? Unfortunately, things are not that easy. To understand why, we need to understand how malicious peers attack the system. A malicious peer, as we outlined in the introduction, is one that offers corrupt or infected resources or poor quality services and/or gives dishonest opinions about other peers. Let us look at a selection of malicious peer types and attacks, most of which come from Marti and Garcia-Molina's "Taxonomy of Trust" Marti & Garcia-Molina (2006): traitors, colluding peers, front peers, whitewashers, and Sybils. A *traitor* is a peer that is initially well behaved in order to build up its trust

value before it begins behaving maliciously. Traitors are hard to defend against because the trust value is a measure of trustworthiness—the probability that the next transaction with that peer will be successful—as opposed to a measure of the peer's intent. One common approach to defending against this type of malicious peer is to make trust difficult to build up but easy to lose. *Colluding peers* work together to subvert the trust mechanism. The simplest form of collusion is for these peers to rate each positively and all other peers negatively. *Front peers* or "moles" behave well but are dishonest in reporting the trustworthiness of the malicious peers with which they are colluding. *Whitewashers* are peers that leave the network and re-enter with a new identity in order to shed any bad reputation. Finally, a *Sybil attack* occurs when a single user creates multiple peers—tens, hundreds, thousands of peers—in the network in order to influence the trust mechanism.

The reason that we cannot simply eliminate low trust value peers from the system is that these malicious peers and their attacks take advantage of the gap between *reliability*—whether a peer provides good resources or services—and *credibility*—whether a peer reports honestly about the reliability of other peers. In most reputation-based trust systems, an offering peer is meant to be judged on its reliability in providing resources but, in reality, its trust value is based on the credibility of the peers asked to give their opinions about its trustworthiness. The result is that honest peers can sometimes have low trust values because malicious peers lie when asked about the honest peers' trustworthiness, while similar lies can create high trust values for malicious peers.

Trying to eliminate the peers with the lowest trust values is also complicated by the fact that *strangers*—new peers in the network—are always given the lowest possible trust value. It would appear to make more sense to give a new peer an average trust value. If it were to act unreliably, the trust value would go down, and if it were to act reliably, the trust value would go up. Unfortunately, the average approach benefits whitewashers. As soon as a malicious peer's trust value gets low enough for the peer to become ineffective, it simply re-enters the network with a new identity and gets an automatic boost in its trustworthiness. Giving the lowest possible trust values to strangers makes whitewashing less lucrative but it also makes it difficult to distinguish new peers from peers that have bad reputations, which is yet another reason why lower trust value peers cannot be excluded from the system, since a new peer would never have the chance to gain trust. One drawback to giving new peers the lowest value is that they are unlikely to be picked probabilistically based on that value. To overcome this problem and allow new peers to gain trust, most trust mechanisms allow for a small percentage of transactions to choose equaling between all offering peers. In doing so, new peers are given the opportunity to gain trust at the cost of a small increase in the transactions going to badly-behaved peers.

The EigenTrust paper (Kamvar *et al.* , 2003), which set the tone for the development of the reputation systems that came afterwards, sets out five design considerations for P2P reputation systems. The system should be *self-policing*, meaning that the peers themselves should run the system rather than defer to a centralized authority. Historically, centralized P2P networks such as Napster, which has a central authority that coordinates transactions, have been vulnerable to legal attacks by industry associations trying to protect copyrighted materials, whereas truly distributed P2P networks such as Gnutella have not. In fact, the legal woes of Napster were reported at the time as the motivating factor in designing Gnutella to be distributed (Barr, 2000). The system should maintain *anonymity* so that a user's identity is protected. The system should not assign any *profit to newcomers*, meaning that strangers should get the lowest trust value possible, as we discussed above, and be made to work for their reputation. The system should have *minimal overhead* so as not to pose an undue burden on its participants' computers or network connections. Finally, the system should be *robust to malicious collectives* so that it can still successfully function even if a large percentage of peers are colluding.

What we want to eliminate with our approach is the need for extreme robustness in the face of malicious collectives. Impressively, the EigenTrust authors claim that their algorithm still generates valid trust values in networks that have up to 70% of peers maliciously colluding. We understand the need to be able to withstand mass collusion and we believe it is an important design consideration for any new reputation system, but we wonder why anyone would willingly join a network where it is possible even to have that many malicious peers. It would be nearly impossible to do e-commerce in such a toxic environment. Our

approach is to create an environment where it is actually beneficial for the participants to ensure that malicious and selfish peers are kicked out of the network. We think this goal is possible using existing trust mechanisms, which themselves have many useful properties. As such, our proposal augments rather than replaces the existing algorithms.

2.3 Related work

We looked at a number of papers that propose trust mechanisms. EigenTrust (Kamvar *et al.* , 2003) is one of the seminal works in the field and is an application of the Google PageRank algorithm to trust in P2P networks. The algorithm assigns to each peer in a P2P network a unique global trust value based on the peer's history of uploads. The trust value is calculated by a distributed and secure method using Power iteration. By having querying peers use these global trust values to select offering peers from whom they download, the EigenTrust algorithm has been shown to effectively decrease the number of inauthentic files in the network. There are a number of papers that try to improve on EigenTrust by adding cyclic partitioning (Abrams *et al.* , 2005), and by extending EigenTrust to work with groups (Ravichandran & Yoon, 2006). On the contrary, we propose a trust by association approach in which the trustworthiness of a peer is measured based not only on the peer's own behaviour, but also on the behaviour of other peers that the peer has invited to join the network.

There are some papers that take different approaches: Androutsellis-Theotokis *et al.* (2007); Kerr & Cohen (2006b); Marti & Garcia-Molina (2004, 2006); Ries (2007); Singh & Liu (2003); Yu *et al.* (2004). Marti and Garcia-Molina (Marti & Garcia-Molina, 2006) present a taxonomy of reputation system components and their properties. Their goal is to organize existing ideas and work in order to facilitate the design of P2P networks. Ries (Ries, 2007) proposes CertainTrust, a trust model that is capable of expressing the certainty of a trust opinion depending on the context of use. Kerr and Cohen (Kerr & Cohen, 2006b) develop a trust framework in an e-commerce environment where selling agents operate with units of trust that are risked when entering into transactions with buying agents. In this system, when buying agents are disappointed, their reporting causes selling agents to lose trust units and thus eventually be unable to engage in transactions. Most of these approaches normalize the trust value produced for a peer to the interval $[0, 1]$. Others, such as the MoR-Trust system (Androutsellis-Theotokis *et al.* , 2007), quantify the trust value as an unbounded value such as a monetary value; that is, one can safely transact with this peer up to $\$x$. Our approach, as detailed in Section 3, introduces a computational method of making peers accountable for one another in a structured manner, hence encouraging the participation of well-behaving peers and the removal of badly-behaving ones. For reasons that we discuss in Section 3.3, our approach only works with those trust mechanisms that generate bounded results.

Breban and Vassileva (Breban & Vassileva, 2002), working with the trust evolution model of Jonker and Treuer (Jonker & Treur, 1999), allow agents to progressively learn which agents are the best ones to join their coalition. Sabater and Sierra (Sabater & Sierra, 2001) present a rich model of reputation, allowing for the integration of various factors being modelled, towards a single reputation rating, to demonstrate that the evaluations can provide significant detail, towards trading decisions. Tran (Tran, 2005) proposes a direct experience trust model that protects buying agents in an electronic marketplace from being harmed infinitely by dishonest selling agents. Yu and Singh (Yu & Singh, 2000) believe that it is useful to take advantage of ratings offered by other agents in a designated neighbourhood, which may improve the evaluation of selling agents by potential buying agents. In addition, other researchers have clarified the importance of modelling reputation, saying that reputation can be used to "impose appropriate constraints" (Jones & Marsh, 1997) or "the formalization of reputation is an excellent place to begin in formalizing trust" (Carter *et al.* , 2002). Yet, none of these researchers has specifically taken advantage of the external social relationship that should exist between agents, an important factor that our model is based on. In particular, the association trust value for a peer x proposed by our model consists of two main components: the base trust value of x itself and the aggregated trust value of other peers invited by x due to some existing external social relationship between x and these peers. This approach provides a mechanism to tie x with the peers x invites to the system, thus motivating x to make good use of its external social relationship to find the best possible peers for the network and to apply social pressure to keep those peers well-behaved after they have joined.

Various researchers have investigated the use of different mechanisms for P2P networks. Papaioannou and Stamoulis (Papaioannou & Stamoulis, 2006) introduce a reputation-based mechanism that assigns better service to better-behaving peers. This mechanism classifies reputation into two categories, provider selection and contention resolution. In provider selection, a peer among the offering peers is chosen to provide the service. In contention resolution, a peer among the querying peers is selected by the provider peer. In Ranganathan *et al.* (2004), the authors propose a reputation-based peer-approved mechanism, which rates peers according to the number of files they are advertising. Peers are allowed to download files only from peers with lower or equal ratings. Although their results show that the proposed mechanism can be used to combat selfish behaviour, this mechanism also allows malicious peers to advertise a high number of corrupted or useless files and still benefit from the system. BitTorrent is a widely used and scalable P2P protocol that adopts the *tit-for-tat* strategy. This strategy enables peers to optimize their download and upload rates. Peers typically upload to the k peers that recently provided them with the best download rates. However, recent studies (Jun & Ahamad, 2005; Thommes & Coates, 2005; Bharambe *et al.* , 2006) show that the tit-for-tat strategy does not effectively reward the well-behaving peers and properly punish the badly behaving ones. In addition, it is possible, under this strategy, that selfish peers can get more bandwidth while honest peers can unfairly receive low download rates. In contrast, our proposed model makes use of the external social relationship that we believe must exist between peers to encourage the participants to invite potentially well-behaving peers to join the network, while actively seeking out and removing badly-behaving peers. As detailed in Section 3, our approach stimulates the growth of the network and takes into consideration the network's maturity. In addition, the proposed approach provides an appropriate level of fairness to new peers—who have just entered the system—and last peers—who join the network after it has reached maturity when the pool of possible new peers is small.

3 Trust by association

We propose a trust system that creates an environment—an economy—that encourages peers to act properly. As discussed in Section 2.2, hard and fast rules cannot be used to exclude malicious and selfish peers from the system, since malicious peers can lie and new peers are necessarily indistinguishable from bad peers. Instead, we must craft a system that encourages good behaviour. We start with the concept of an invitation-only P2P network where existing peers, *inviters*, ask new peers, *invitees*, to join the system. By tying an inviter's trust value to the trust values of its invitees, which we call *trust by association*, we intend to make the inviter responsible for the behaviour of its invitees. It is our contention that, as a rule, an inviter will not invite a total stranger to join the network and, therefore, there must be some relationship between the inviter and invitee that is external to the network. The invitee may not be the inviter's close friend, but there will be some common external connection such as mutual friends, membership in the same club, or participation in the same on-line community. We hope to take advantage of the external pressure that can be brought to bear using this external relationship in order to make it difficult for malicious and selfish peers to abuse the network without consequence, and to give incentive for both good behaviour and network growth.

3.1 Basic assumptions

We make some basic assumptions about the P2P network: It is invitation only which creates the external social network on which our approach relies, it is created to trade a specific type of resource or service, it uses an existing, reputation-based trust mechanism that produces bounded trust values, and it has a central point of calculation.

Invitation-only network: As we discuss above, our approach relies on the external social relationship that we believe must exist between an inviter and an invitee in an invitation-only network. The external relationship allows peers to bring social pressure ("peer" pressure) to bear on badly behaving peers. Peer pressure provides a subtler mechanism for enforcing good behaviour than the coarser ability of simply removing a peer from the network.

Homogeneous resource or service: We assume that the network is created to trade a specific type of resource or service where the prices of the resources or services do not vary widely in price (e.g. iTunes offers music, videos, movies, and games at relatively the same price). We make this assumption to level the playing field in the external social network. Everyone has friends that have different tastes or means. It would be unfair to base the trust value of one person who purchases antiques for thousands of dollars on the trust value of their friend who buys video games for tens of dollars, as is possible on more general trading networks that function like eBay. We can eliminate this bias by restricting our approach to networks with homogeneous offerings.

Bounded existing trust mechanism: We require a bounded trust mechanism, such as EigenTrust (Kamvar *et al.* , 2003), rather than an unbounded mechanism, such as MoR-Trust (Androutsellis-Theotokis *et al.* , 2007), for the same reason as in the previous assumption: we want to level the playing field in the external social network. Bounded trust mechanisms tend to focus on volume rather than the value of transactions. In a truly homogeneous network where all the offerings have the same value, there should be no substantive difference between bounded and unbounded trust mechanisms. However, if the network is only semi-homogeneous, like iTunes where the values/prices are comparable but not the same, the bounded results provided a fairer comparison between peers.

Central point of calculation: In order to simplify the presentation of our approach, we assume for the moment that all calculations are made at a central point. We loosen this assumption later in our discussion of how our approach can be decentralized, or at least made scalable, in Section 3.5.

3.2 Invitation-only P2P network

Invitation-only P2P networks are the basis of trust by association. Although there is generally some cost in time, effort, or money to join most P2P networks—the cost being used as a deterrent for whitewashers—generally, anyone can join. We choose to be more selective in only allowing participants to join our network as peers if they are invited to do so. We assume that there is secure mechanism to ensure that only invited peers participate in the network, the functioning of which is outside the scope of this work. There is precedent for an invitation-only network. Gmail started very successfully as an invitation-only web application before being opened up to the general public. It had a rudimentary trust system where the longer a participant was on the system without bad behaviour, such as spamming, the more people she was allowed to invite to join. These types of networks grow geometrically and mature quickly. We discuss the growth and maturity of the network in more detail in Section 3.6.

The invitations create a structure in the normally unstructured environment of P2P networks. Single invitations create a tree structure where the inviters and invitees become the parents and children, respectively, in the tree. In the rest of the paper, we use these terms—parents/inviters and children/invitees—and their singular forms interchangeably. We discuss the benefits and drawbacks of single and multiple invitations in Section 3.4.1. As we see in Figure 1, the root of each branch of the tree is an *initial peer* in the network, which sits under a "theoretical root". The initial peers of any P2P network are almost always the creators of the network and their close friends or associates. We can safely assume that these initial peers are trustworthy and that the people they invite to join the network will most likely also be trustworthy and well-behaved. In fact, EigenTrust relies on the trustworthiness of these initial peers to protect the algorithm from attacks.

We also rely on the trustworthiness of the initial peers but in a different way. The practical consequence of the trustworthiness of these first peers, the initial peers and their first invitees, is that the next set of peers invited into the network can only be of equal or less trustworthiness. It means that the first malicious or selfish peers that are added to the system must be added by a more trustworthy peer. This fact gives us leverage. If we can make the badly-behaved peer "cost" something to the well-behaved peer that invited it, we can make it worthwhile for an inviter to "uninvite" the invitee. We deal with how to remove badly behaving peers in Section 3.7. We express this cost in a lower trust value which can in turn be converted into a real-world cost by tying it to something valuable in the network, such as the monetary cost per transaction

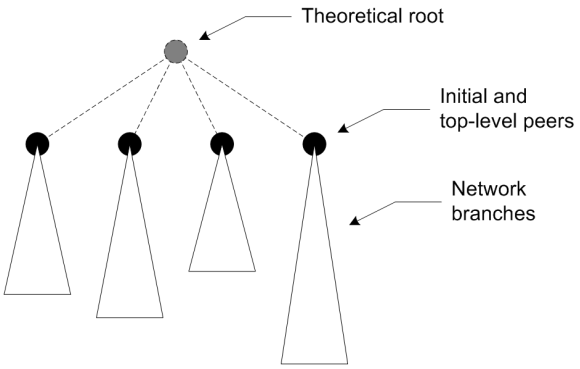

Figure 1: Network structure created by invitations.

or the download speed (download speed would be especially effective for large files like movies if the variance in speeds for high and low trust values were significant enough). Essentially, the better behaved the inviter and her invitees are, the lower the cost for her to participate in the network.

3.3 Basic equation

We assume that the network implementors have chosen a bounded trust mechanism, such as EigenTrust (Kamvar *et al.* , 2003). Let $T(x)$ be the *base trust value* for a peer x based on the chosen trust algorithm, where $T(x)$ is bounded in general. Without loss of generality, our examples are based on a file-sharing network where $T(x) \rightarrow [0, 1]$. Trust by association requires that a peer's base trust value be combined with those of its invitees. We call the resulting value the *association trust value* since it is derived from the association amongst these peers. We replace the base trust value with the association trust value when deciding on which offering peer will be chosen to fulfill a query. Let $A(x)$ be the association trust value for a peer x. The simplest way to calculate $A(x)$ is to combine x's base trust value with the average base trust value of its children. Let C_x be the set of peers that have been invited by x to join the network. Let $c_x = |C_x|$ be the cardinality of set of children. Equation 1 shows our basic starting equation.

$$A(x) = (1 - \alpha)T(x) + \alpha \frac{\sum_{y \in C_x} T(y)}{c_x} \tag{1}$$

The value α represents the *association cost* for x to associate with C_x, where $0 \leq \alpha \leq 1$. We initially set $\alpha = 0.5$, so that half a peer's association trust value comes from its own base trust value, which is a measure of its own behaviour, and the other half comes from the peer's children. For the sake of simplicity, we define the second term to be 0 when $c_x = 0$ to avoid division by 0.

The first thing to note about Equation 1 is that by using the children's base trust values rather than their association trust values, x is made responsible only for its children's behaviour and not for their choice of whom they invite to join the network. We use the base trust value by design. There are a number of problems with using the association trust value for the children. Using the association trust value makes the calculation of $A(x)$ recursive, since we must calculate $\sum_{y \in C_x} A(y)$, which necessitates the calculation of $\sum_{z \in C_y} A(z)$ and so on until the leaf nodes of the tree are reached. We refer to this form of the equation as the *recursive approach*. The higher x is in the tree, the exponentially larger the number of peers' base trust values we must combine in order to calculate $A(x)$. Using the base trust values means that we need only have the base trust values of the peer and its children, which should make distributed calculation possible and should make the algorithm scalable to larger networks.

There is an intangible but much more serious consequence to using the association trust value for the children. The recursive nature of the calculation will propagate the low trust values of new, malicious, and selfish peers up the tree, far past the peer that brought these peers into the system. It is unfair to hold all these peers responsible for the actions of invitees far down the tree. We discuss exactly how far trust values should propagate in Section 3.3.1. The recursion also makes an effective attack. If a peer creates or invites multiple levels of malicious peers, it can raise the cost to use the network for every peer in its branch of the tree. Our approach avoids the recursion problem but does allow for a front peer to "hide" malicious peers. We address the hiding of peers in Section 3.3.2.

Our basic equation is only a starting point. Let us assume for the moment that a peer's goal in the network is to maximize its association trust value and, therefore, minimize its cost. It is easy to see that a peer can maximize its value from Equation 1 by inviting a single peer to join the network. As long as that peer quickly reaches a base trust value that is no worse than the inviting peer's base trust value, there will be little or no cost to the inviting peer. In fact, if the base trust value of the invited peer is better than the inviting peer's, the inviting peer may actually benefit from the invitation. There is no incentive, however, to invite more than a single peer to join the network, since doing so is risky. Eventually, someone is going to invite a malicious or selfish peer to join. If a peer can guarantee that the one peer they invite will be well-behaved, there is no need to risk inviting others. If the invited peer turns out to be malicious or selfish, it should be easy enough to "uninvite" them and find another trustworthy invitee. We need to encourage multiple invitations or we face the prospect of our network having linear growth. The goal for trust by association is to facilitate e-commerce using the P2P model. E-commerce in these networks requires a critical mass of well-behaved participants in order to both feed the supply and maintain the demand on the network. We are unlikely to reach this critical mass with linear growth. Imagine how unsuccessful eBay would have been with linear growth. The buyers would not necessarily want what the sellers were selling. Put another way, our user would not stick around for months on the network in the hopes that someone else will show up with the same interest in her favourite TV show. The consequence of linear growth is a network that will not offer enough choice to its participants for them to invest time in helping the network grow. Under these circumstances, the network will not be able to grow fast enough to sustain itself and it will fail. We need to modify our equation to encourage invitations. We discuss how this encouragement might work in Section 3.4.

In the last paragraph, we assumed that the peer's goal in the network is to maximize its association trust value. The obvious approach to doing so is for a peer to base its actions on the actual equations used to calculate the value. While this assumption simplifies our analysis, making two actions comparable by their effect on the peer's trust value, we think it is short-sighted. In practice, only a handful of peers will take the time to analyze the calculation of association trust value or base trust value to determine the best course of action, and many of these peers will be malicious peers looking to maximize their opportunities for malice. In practice, peers are more likely to exchange tips in on-line communities or develop rules of thumb on how to get the most from the system. While these approaches may be useful to the users, they are unlikely to produce optimal results.

3.3.1 Extent of responsibility

We need to determine the extent of a peer's responsibility—its sphere of influence within the network. Formally, we say that a peer x is *responsible* for peer y, if y's base trust value, $T(y)$, is included in the calculation of x's association trust value, $A(x)$. We give a responsible peer the authority to seek the removal of a peer for which it is responsible as described in Section 3.7. We believe that it would be unfair for a peer to be affected by the behaviour of another peer in the network and not to have any recourse. We have already looked at two variations of how invitees' base trust values can be included in a peer's association trust value. In Equation 1, the peer is only responsible for its invitees. In the recursive approach, the peer is responsible—by our definition—for all peers below it in the tree. We know that the recursive approach is not desirable for the reasons we noted above. When we add responsibility, we believe that it is not in the best interests of the network to give peers the ability to remove *any* peer lower than they are in the tree.

We propose that the best approach is to limit a peer's responsibility to its invitees, as in our basic

equation. We want to take advantage of the informal social network used by people to get invitations to join the network as peers. The same social network allows peers to exert external pressure on badly-behaving peers. We suggest that it only makes sense to limit a peer's responsibility to those peers that fall within its external influence. It is safe to say that a peer's invitees fall within its external influence since it must have had some external communication in order to know to invite them to join. We might also add those peers invited to join by the peer's invitees (that is the invitees of the invitees), but we would argue against doing so. It is short-sighted to assume that everyone that a peer invites is a close friend. In fact, there can be several levels to the social network that leads to the invitation. A peer could be asked to invite the friend of a friend over whom the peer would have little influence. If that invitee also invited a friend of a friend to join, the original peer would have no external relationship at all with the new peer and its influence would be limited solely to the power of removal, a very coarse tool. The other possibility is to extend a peer's responsibility to include the peer that invited it, allowing for limited upward feedback in the network. Including the parent would make it difficult for a malicious peer to invite well-behaved peers to help it camouflage its malicious activities. However, we believe that including the parent is fundamentally unfair to the peer. The peer cannot be expected to know the status of the inviter before joining the network nor to know what that status means. It is unfair to punish a peer for choosing to accept the invitation from a friend who turns out to have a low trust value.

3.3.2 Hiding badly behaving peers

Limiting the extent of a peer's responsibility to its invitees has the unfortunate side effect of allowing a peer to easily hide badly-behaving peers in the tree below it. In order to avoid the suspicion of its parent and act as a front peer, the peer must remain well-behaved but is free to invite as many malicious peers as it wants into the network. The only cost is to its association trust value. If this cost is undesirable to the front peer, it can set the malicious peers off one level by inviting a Sybil of itself or another front peer. Even though the recursive approach would make this type of attack more obvious, we assert that the benefits do not outweigh the drawbacks that we have already discussed.

We turn to the social nature of the network for a solution to this problem. A peer could easily launch a Sybil attack—creating hundreds or even thousands of copies of itself—if we assume that the base trust algorithm being used imposes only a minimal cost to the creation of new peers, where the cost is the effort required to join the network as a new peer (some trust algorithms require a signed public key certificate issued by a recognized authority in order to join the network, making large-scale Sybil attacks extremely costly). These new peers must be invited to join the network. The peer or its invited front peer could invite them but it would be concentrating them under itself. We discuss how to deal with large concentrations of badly-behaving peers in Section 3.7. The only other option is for these peers to be invited individually to join by peers that are widely dispersed in the network. We have already noted that in an invitation-only network a malicious peer must be invited to join by a well-behaved peer. The attack would only work if a large number of well-behaved peers could be convinced to invite malicious or front peers to join the network. We believe it is unlikely that the external social network of the attackers would be large enough or well-behaved enough to make this attack viable. Regardless, we have greatly increased the cost of such an attack.

3.4 Encouraging invitations

We know that a peer can maximize its association trust value using Equation 1 by simply inviting a single well-behaved peer to join the network. We also know that a single invitee per peer leads to the linear growth of the network and a very good chance that the network will not grow large enough, fast enough to make the network viable. We are limited in our approaches to this problem by the fact that we are only combining existing base trust values. We have already limited the responsibility of a peer to its invitees for the reasons that we discussed in Section 3.3.1. We must now look at how we can use the trust values we have to encourage a peer to invite more peers to join the network and get the geometric growth that will allow the network to grow and prosper.

Our approach is to allow peers to keep more of their base trust value proportional to the number of peers they invite to join the network. We know that the first peers in the system are by definition more trustworthy than the peers that come after them. Although the situation may reverse at some point, we believe that rewarding peers with more emphasis on their own behaviour will act as encouragement for all well-behaved peers to invite peers to join the network. At the same time, if some of these invitees are malicious or selfish, we need to ensure that the peer that invited them is motivated to remove them.

We address the proportional element by changing how we calculate the association cost. Instead of it being a constant—$\alpha = 0.5$ in our initial discussion of the basic equation—we replace α with modifiers based on the number of children. To make it work, we must define a maximum number of children, c_{max}, but luckily there are additional motivations for having such a maximum that we discuss in Section 3.6. Let max be a function that returns the maximum of the values given to it (similarly, min for minimum values). We use max to make sure that the result stays within bounds. Here is our new equation:

$$A(x) = \left(\frac{\mathbf{min\{c_x, c_{max}\}}}{\mathbf{c_{max}}}\right) T(x) + \left(\frac{\mathbf{max\{c_{max} - c_x, 0\}}}{\mathbf{c_{max}}}\right) \frac{\sum_{y \in C_x} T(y)}{c_x} \tag{2}$$

The obvious problem with this form of the equation is that x's association trust value is 0 when it has no invitees, and that the invitees have no impact on its association trust value when it has the maximum number of invitees. An association trust value of 0 means that the peer is unable to participate in the network since it can never be chosen as an offering peer for a transaction. In systems where in a small percentage of transactions the offering peer is chosen entirely randomly rather than probabilistically based on its trust value, the peer may still be chosen but there is no effect on its association trust value. The peer is made selfish by definition until it invites someone to join. We feel the result is an unrealistic expectation placed on the peer and one that makes the network unsustainable since there will always be new peers that are unable to act. With geometric growth, it is possible that these forced-selfish peers will equal or exceed the number of fully participating peers. Therefore, the peer should always be able to participate in the network to some extent without children. The problem at the opposing end of the spectrum is that the invitees have no impact on the peer when the peer reaches its maximum children. The peer is no longer responsible for them since they are no longer included in the calculation of its association trust value. The invitees are essentially set free in the network and can behave as they like. Neither extreme is desirable.

We address the problem with the extremes in the number of children by bounding the modifiers in the interval $[b_{min}, b_{max}]$ where $0 < b_{min} < b_{max} < 1$. Let $b_{int} = b_{max} - b_{min}$ be the interval between the minimum and maximum bounds of the modifiers. Tuning the values of b_{min} and b_{max} depends on the base trust algorithm used and the nature of the network and should be based on experimentation. Here is our updated equation:

$$A(x) = \left(\mathbf{b_{min}} + \mathbf{b_{int}} \frac{min\{c_x, c_{max}\}}{c_{max}}\right) T(x) + \left(\mathbf{b_{min}} + \mathbf{b_{int}} \frac{max\{c_{max} - c_x, 0\}}{c_{max}}\right) \frac{\sum_{y \in C_x} T(y)}{c_x} \tag{3}$$

The resulting equation has the properties which we seek. It rewards the peer for inviting peers to join the network but still allows a new peer to participate based on its own behaviour. The peer is able to benefit fully by inviting at least one peer to join the network and is able to insulate itself from that peer to some extent by inviting more. The only time that inviting peers will not be attractive is when the peer knows that the invitee will be more trustworthy under the base trust algorithm than it itself is. Our hope is that our approach will keep the network clean enough of malicious and selfish peers that having peers in this situation will be rare.

3.4.1 Single and multiple invitations

The structure of the network we have been discussing is based on a single invitation being needed from an existing peer in order for a new peer to join the network. A variation on single invitations is requiring multiple invitations, perhaps 2 or 3, before a new peer can join. Multiple invitations raise the joining cost for a new peer, since it has to find more than one peer to "sponsor" it onto the network. A higher joining cost means that it is more difficult for malicious peers to whitewash or launch Sybil attacks. However, a higher joining cost also means slower growth for the network and complicates the network's administration.

One of the benefits of single invitation is the creation of separate branches rooted in the initial peers. We use these branches in the administration of the network that we describe in Sections 3.5 and 3.7. Allowing multiple invitations creates many parents, potentially from different branches. Multiple parents have no effect on the association trust equations we have developed, if we allow that each of the parents is equally responsible for the invitee and can call for its removal. Allowing for removal in this way opens up the possibility of one parent attacking another by removing well-behaved rather than badly-behaved peers in the hopes of having a bigger negative effect on the other parent. We address the problem of removing well-behaved peers in Section 3.7.

The bigger problem is that a peer's parents may come from different branches and complicate the administration of the network by essentially mixing all the branches together into a single, interconnected graph. We would like to keep the branches separate and there are a number of ways that this separation could be accomplished. We could require that all invitations for a peer come from peers in the same branch. We feel, however, that such a rule is unworkable because some branches will naturally be smaller than others. Another approach is to allow invitations from multiple peers, but with only one of those peers becoming responsible for the new peer. However, we would still need an incentive for the peers that do not become the primary peers. Perhaps the best approach is to assign the responsibility randomly to one of the peers, although this approach raises the concern that some inviters would be probabilistically "starved" of new peers. Another approach would be to offer some sort of incentive for the non-responsible peers making the invitation, such as a monetary bonus in their network bank account. The reward would have to be carefully set to reward invitations without making them profitable and, thereby, a target for attack. The result either way of choosing a single responsible peer from among multiple inviters is to allow the invitation trees to remain separate. Although, these approaches are definitely possibilities, we believe that the single invitation structure is more understandable to the peers, more feasible to implement than the other options, and that there are more effective ways to raise the joining cost of the network.

3.5 Decentralization

In Section 3.1, we made the assumption that there is a central point of calculation in order to simplify the presentation of our approach. We loosen that assumption here with a sketch of how our approach might be decentralized and made scalable. We start by making the distinction between the operation of the network and its administration. We define the operation of the network as the activity surrounding transactions between peers: querying, offering, and the calculation of trust values. We define the administration of the network as the adding and removing of peers. In our approach, the operation of the network can be completely decentralized, while the administration of the network can at the very least be made scalable.

The only operation of concern to us is the calculation of trust values for peers. The other operations—querying and offering—function as they would in any other network. A query is sent out to some or all of the peers in the network and those peers with the requested resource or service reply with an offer. The querying peer must now retrieve the association trust values of the offering peers. We start by assuming that the underlying trust mechanism that is used to calculate the base trust value is decentralized. To decentralize the calculation of association trust values, we look to the concept of score managers (as in EigenTrust). A score manager is a peer in the network that holds the trust value for a unrelated peer. It is possible that any given score manager could be malicious, so a peer's trust value can be held by multiple unrelated score managers. Assuming that our attempt to bound the number of children for each peer (c_{max}) is successful, the score manager or managers for a particular peer need only gather the base trust value for a constant number

of peers. Depending on the nature of the network, the score manager can calculate the peer's association trust value on demand, or, if it is possible, it can update the score each time one of the peers involved in the calculation completes a transaction.

The administration of the network is not as straightforward. We need to ensure that when a peer is invited into the network the relation between it and its inviter is maintained. Although it may be possible to distribute this function to the score managers, it should be a relatively infrequent enough event that a more centralized approach can be used. We already have peers that are known to be trustworthy in the initial peers. We propose that each branch under an initial peer will have its own branch authority. The branch authority would be responsible for maintaining the authoritative record of the structure of the branch. Because of its inherent trustworthiness, it would also have a role in the removal of peers from the network as discussed in Section 3.7.

Obviously, restricting the number of branches, and branch authorities, to just the initial peers is not scalable. So, when a branch becomes too large, a new branch can be created by breaking off a sub-branch at the top of the tree whose new root is a trusted peer that is promoted to be a "top-level" peer. (Similarly, the branch authorities can be centrally hosted at the start of the network and can be distributed to separate hosts as the network grows.) Care must be taken in promoting peers, since no peer is responsible for these top-level peers. It is possible that a malicious peer, especially a traitor (perhaps a friend with whom the network's creators have had a falling out), could become a top-level peer. In order to protect the network, we must allow for the removal of top-level peers. If we think of the theoretical root of the network as the collection of all top-level peers then all top-level peers through this root would be collectively responsible for each other. If a majority or set number of top-level peers agree, they could remove a top-level peer using the rules outlined in Section 3.7.

3.6 Growth and maturity of the network

It is easy to see that if we are successful in encouraging multiple invitations per peer then the network will grow geometrically. Eventually, the growth must slow because the network has saturated its market (for instance, a network trading Lawrence Welk reruns might saturate the market of tech-savvy octogenarians quite quickly). We must deal with both the network's growth and its maturity. We must not assign too much trust to the first peers that join the network because they have a bigger pool of potential invitees to draw from and we must not freeze out the last peers to join because of an inability to find others to recruit to the network.

The problem of first peers is addressed by the maximum number of invitees that we introduced in Section 3.4. By restricting peers to a maximum number of invitees, we ensure geometric growth by allowing multiple invitees but also encouraging multiple levels of invitees. There is one problem, however, that we did not address earlier. We need to deal with the situation where a peer invites more than c_{max}. Right now, the average of all the children is taken regardless of their number. We want to discourage peers from inviting more than the maximum, so we propose to take the average of the c_{max} children in C_x with the lowest base trust values. The change makes it impossible for the peer to benefit from inviting more than c_{max} peers to join. Let *cmin* be a function that returns the c_{max} peers with the lowest base trust values from the set of invitees C_x if $c_x > c_{max}$ and returns c_x peers, otherwise.

$$A(x) = \left(b_{min} + b_{int}\frac{min\{c_x, c_{max}\}}{c_{max}}\right)T(x) + \left(b_{min} + b_{int}\frac{max\{c_{max} - c_x, 0\}}{c_{max}}\right)\frac{\sum_{\mathbf{y}\in\mathbf{cmin(C_x)}} T(y)}{\mathbf{min\{c_x, c_{max}\}}} \quad (4)$$

We have already addressed the problem of last peers by allowing a peer to maximize its association trust value with a single invitee but still allowing it to have a trust value of up to b_{min} if it has no invitees.

However, there may be networks where the administrators would like to have a minimum number of invitees larger than 1 before a peer can maximize its trust value. In other words, it may be beneficial to require peers to invite more than one peer before they can maximize their association trust value. Let c_{min} be the minimum number of invitees before a peer is able to maximize its association trust value. Here is our final equation:

$$A(x) = \left(b_{min} + b_{int}\frac{min\{c_x, c_{max}\}}{c_{max}}\right)T(x) + \left(b_{min} + b_{int}\frac{max\{c_{max} - c_x, 0\}}{c_{max}}\right)\frac{\sum_{y \in cmin(C_x)} T(y)}{\mathbf{max\{c_{min}, min\{c_x, c_{max}\}\}}} \quad (5)$$

Equation 4 is equivalent to Equation5 when $c_{min} = 0$.

Our final equation has all the features that we have been discussing. It allows for the growth of the network and for its maturity when new peers will be hard to find. There is no reason that the parameters need to be static for the life of the network. We would encourage the administrators of the network to tune the variables to reflect the best interests of the network at whatever stage of growth it is in. For instance, a low b_{min} and high c_{min} can be used to encourage invitations at the start of the network. These variables could be raised and lowered respectively as the network reaches maturity, to encourage peers that are unable to find invitees to stay in the network. Finally, we suggest that c_{max} should remain constant, since lowering it later in the growth of the network will essentially punish the peers that have maximized their invitees.

3.7 Removing badly behaving peers

Removing badly behaving peers from the network is the most difficult and most important task of our approach. We need to ensure that removal is not misused. If improperly handled, the ability to remove peers could easily be used by malicious peers to attack the system. There are a number of problems with which we must deal. We need to make sure that peers do not have the power of removal themselves. It should be administered by a third party to ensure that the removal is warranted. We need to guard against peers that selfishly try to maximize their association trust value by removing otherwise well-behaving peers because they fractionally draw down the peer's trust value. We need to deal fairly with the invitees that are "orphaned" by the removal of their inviter.

The obvious place to put the power of removal is in the branch authority, since the peer and its invitee must be in the same branch under the rules we discussed in Section 3.4.1. The peer applies for the removal of its invitee. The removal itself can either be automatic or moderated by the human in control of the branch authority. Automatic removal is probably preferable because human moderation may not be timely. A removed peer is free to rejoin the network, if it can convince another peer to invite it. If re-entering peers were to keep their history when they rejoined the network, there would be ample incentive for them to whitewash. To encourage peers to keep their identities, they should re-enter the network as new peers. At the same time, the new inviter should be able to review the history of the removed peer. In doing so, we hope that external pressure can be used to correct the peer's behaviour. An inviter who knows the peer's history may negotiate the peer's behaviour before inviting it to rejoin. The policy is focussed on well-meaning peers that have behaved badly rather than malicious and maliciously-selfish peers, who will just whitewash in the same situation.

We want to guard against peers that selfishly try to maximize their association trust value by dismissing well-behaved but "under-performing" peers. We can avoid this situation by setting a network minimum association trust value, a_{min}, above which peers cannot be removed. Since each network is different, we would suggest setting a_{min} to a value that is some constant or percentage below the average association trust value of the network as a whole. Setting a_{min} this way gives us two benefits. Well-behaving peers will be able to remain in the network even if they have somewhat below average trustworthiness due to a lower

than average volume of transactions. At the same time, malicious peers trying to take advantage of the protection afforded by a_{min} will behave well at least some of the time. The network's administrators can tune a_{min} over time to reflect the network's nature. We assume that a peer will not be motivated to remove a newly invited peer that has yet to build up a reputation in the network. Even so, it may be worthwhile to allow for a grace period during which a newly invited peer cannot be removed. The grace period would have to be finely tuned to avoid it being taken advantage of by malicious peers.

We must now deal with the invitees that are "orphaned" by the removal of their inviter. The higher up in the tree that a removal takes place, the more peers that will be affected by it. We want to maintain the social network on which we base our approach. Our best bet for doing so is to assign the removed peer's invitees to its parent since the parent is the only peer that may have some remote external connection to them. However, we believe that the parent would be unfairly punished in this case since the badly-behaving peer may have badly-behaving invitees, which would lead to a lowering of the parent's association trust value, especially if both peers, parent and removed peer, have c_{max} invitees. This approach would be a disincentive to behaviour that we are trying to encourage. We propose a simple scheme to handle the removal. We want to introduce the least disruption possible to the network. We place in order the removed peer's invitees by association trust value and choose a peer randomly from the top half. Alternatively, if the process is not automated, we can let the top half of peers vote to choose one of their number thus increasing the likelihood that the peer is well-connected socially to its peers. This peer is promoted into the removed peer's place and it assumes the rest of the removed peer's invitees as its own in addition to any that it already has. Although this approach is similar to assigning all the invitees to the parent, we believe that a peer's invitee is much more likely to have external contact with other invitees than its parent would. If removal happens early enough in the life of that part of the network, the result will be a windfall of invitees for the elevated peer.

The windfall of invitees may not be beneficial, especially if the peer that was removed was the front peer for a large collective of malicious peers. The responsible peer (the removed peer's parent) would then be faced with removing each malicious peer that gets promoted after each removal. We need a mechanism for dealing with this situation so that a peer is not unfairly penalized. In this situation, the malicious peers have not made the effort to disperse themselves through the network. We propose two solutions. The first comes from the peer itself. A responsible peer contemplating the removal of a peer will be stuck with one of its invitees, so it only makes sense to allow a peer to see not only the base and association trust values of its invitees, which it must be able to see anyway to make the decision to remove the invitee, but it should also see the values for their invitees. If the peer notices that its invitee's invitees are malicious or have a high base trust value and a low association trust value then the peer can report the potential malicious collective to its tree's central authority. A human at the branch authority can then investigate and decide the extent of removals that are needed to deal with the collective. The second solution comes from the first. The branch authority need not wait for a peer to report suspicious activity. Instead, it can pro-actively look for suspicious activity. We suggest that the branch authority should only deal with groups of peers and leave the detection of individually poorly-performing or badly-behaving peers to their inviters.

4 Analysis

We analyze trust by association first by looking at the characteristic curves of each of the equations. We look at the characteristic curves of all our equations to show how each reacts in a number of static scenarios. Our analysis shows that Equation 5 has all the properties discussed above and, even without the benefits of social pressure, is likely to lead to the removal of badly-behaving peers from the network. We then present the results of a simple simulation comparing the EigenTrust algorithm to a trust by association application of EigenTrust. The simulation shows that trust by association based on Equation 5 and using a simple drop rule performs better than EigenTrust alone.

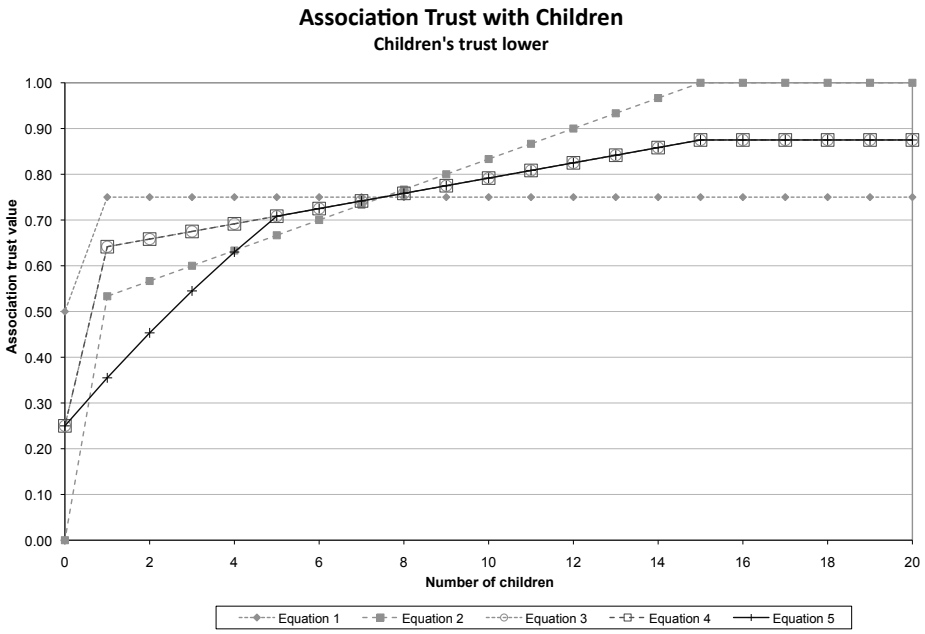

Figure 2: Association trust values for a peer x with varying numbers of children when x is *more* trustworthy than the children. Settings: $T(x) = 1.0$, $T(children) = 0.5$, $b_{min} = 0.25$, $b_{max} = 0.75$, $c_{min} = 5$, $c_{max} = 15$, $\alpha = 0.5$.

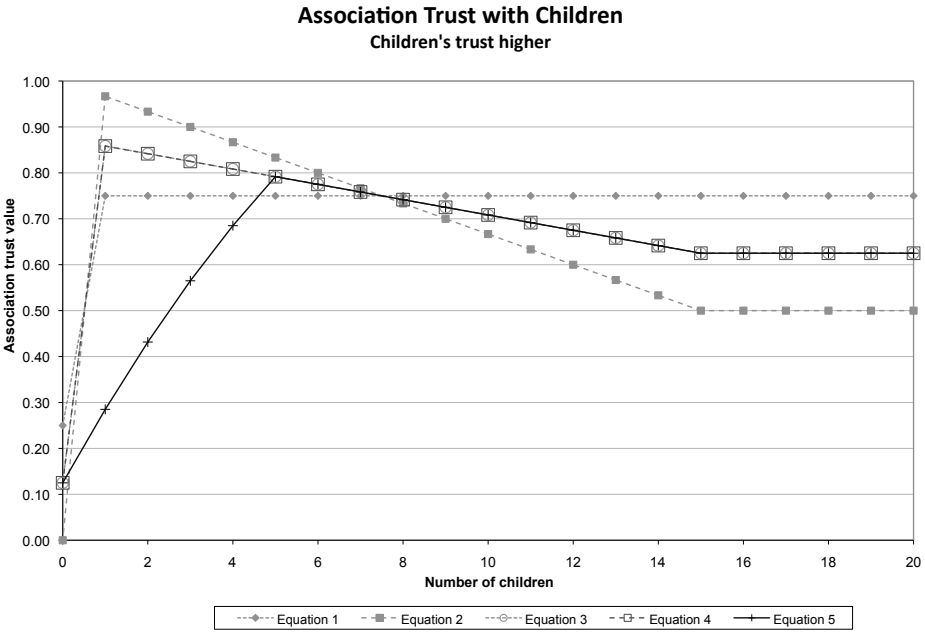

Figure 3: Association trust values for a peer x with varying numbers of children when x is *less* trustworthy than the children. Settings: $T(x) = 0.5$, $T(children) = 1.0$, $b_{min} = 0.25$, $b_{max} = 0.75$, $c_{min} = 5$, $c_{max} = 15$, $\alpha = 0.5$.

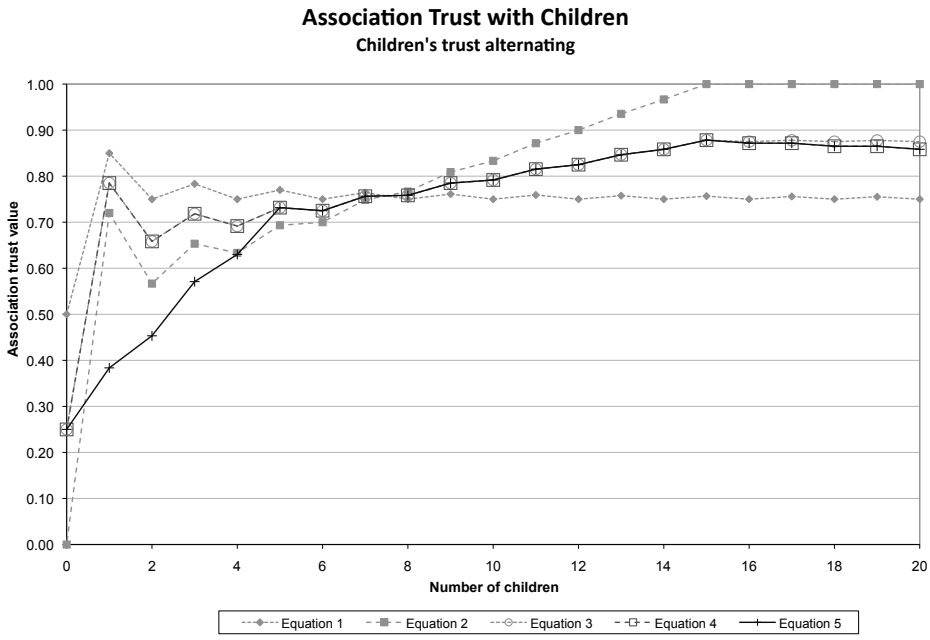

Figure 4: Association trust values for a peer x with varying numbers of children when x is more trustworthy than the children. Each child added alternates between a good trust value (0.7) and a bad trust value (0.3). Settings: $T(x) = 1.0$, $T(odd\_children) = 0.7$, $T(even\_children) = 0.3$, $b_{min} = 0.25$, $b_{max} = 0.75$, $c_{min} = 5$, $c_{max} = 15$, $\alpha = 0.5$.

4.1 Characteristic curves

We present the characteristic curves of each of the equations based on a number of static scenarios. Figure 2 shows the effects of a peer inviting less-trustworthy peers, compared to itself, into the network. Figure 3 shows the effects of a less-trustworthy peer inviting more-trustworthy peers. Finally, Figure 4 shows the effects of inviting a mix of less-trustworthy peers. In reality, when a peer is added to the network, it starts with a 0 trust value and over time builds up to a stable base trust value based on its transactions in the network. We show the effects of this process in the example in the next section. For the moment, we focus on the effects of adding children with static base trust values.

Each of the scenarios shares some basic parameters. We assume that the range of the base trust value is $[0, 1]$ and, as a result, so is the range of the association trust value. The minimum amount that x can keep of its trust value is $b_{min} = 0.25$. The maximum amount is $b_{max} = 0.75$. The minimum number of children that x needs to invite before it can fully benefit from them is $c_{min} = 5$. The maximum number of children is $c_{max} = 15$. Finally, the association cost for Equation 1 is $\alpha = 0.5$.

Each of the scenarios has specific parameters for x and its children. In Figure 2, x is more trustworthy than its invitees, so that its base trust value of x is $T(x) = 1.0$ and the base trust value of each child added is $T(children) = 0.5$. In Figure 3, the situation is reversed with x's base trust value being $T(x) = 0.5$ and the base trust values of the children being $T(children) = 1.0$. In Figure 4, we show the effects of x adding a mix of children that are less-trustworthy with its base trust value being $T(x) = 1.0$, the base trust values of the odd-numbered children being $T(odd\_number) = 0.7$ and of the even-numbered children being $T(even\_number) = 0.3$. Each equation is evaluated for x having from 0 up to 20 children.

4.1.1 Equation 1

The premise behind Equation 1 is that some portion of the parent's association trust value should be its own trust value and the remaining portion should be the average of its children. We surmised in our discussion in Section 3.3 that Equation 1 would mean that the parent would only have incentive to invite a single peer into the network and this conclusion is borne out by the results in the figures. We see in Figures 2 and 3 that there is no benefit to x after it has added its first peer. The concept of benefit here is based on the assumption that the peer is trying to maximize its association trust value. Of course, the situation in the two figures is artificial since we are assuming that each of the children has exactly the same base trust value. We get an idea in Figure 4 of what happens when that is not the case. Since we have specified that the odd-numbered children have a higher base trust value (0.7), as predicted x maximizes its association trust value after adding the first peer.

4.1.2 Equation 2

Equation 2 attempts to encourage invitations to the network by allowing a peer to keep more of its base trust value by adding more children. Figure 2 shows that this approach is effective in minimizing the effects of inviting relatively less-trustworthy peers to join the network. In Equation 1 adding even a single low-performing peer risked lowering the inviter's association trust value. With Equation 2 this risk is lowered because the more children there are the less their collective negative (or positive) effect. The result is shown clearly in Figure 4, where the effects of the even-numbered children's lower base trust value (0.3) has decreasing effect as more children are added.

Figure 3 tells a slightly different story. Here we have x inviting relatively more-trustworthy peers. The result is a lowering of x's association trust value as the children have decreasing influence. The difference between the first two scenarios means that there is a positive incentive for well-behaved peers to invite new peers into the network, even if they are not as well-behaved, and there is a negative incentive for badly-behaved peers to invite new peers. If we assume that peers always act to increase their trust value then this trend means that branches with badly-behaved peers are likely to grow more slowly than branches with well-behaved peers. Of course, this assumption may not apply to collectives of malicious peers that are actively trying to manipulate the network.

There are problems with Equation 2 that can clearly be seen at the extremes of the number of children in Figure 2. When the peer has no children, it has a 0 association trust value regardless of its own base trust value. The peer is made selfish by definition, since a peer that remains childless gains no benefit from offering its resources or services to the network. On the other extreme, there is a problem with having too many children. Remember that we defined a peer x as being responsible for another peer y when y's base trust value is included in the calculation of x's association trust value. Notice that when the number of children reaches c_{max} the children's base trust values no longer have any effect on x's association trust value. We can now see why we defined responsibility the way we did. Even though x may be technically responsible for its children—because their base trust values are input into x's association trust calculation—once the number of children reaches c_{max}, x can safely ignore its children because their behaviour no longer has any impact.

4.1.3 Equation 3

Equation 3 attempts to fix the problems at the extremes of the number of children by introducing bounds on the effect of the children's base trust value. We see the results of the bounds in Figure 2. Instead of having a 0 association trust value when it has no children, x can now have an association trust value up to b_{min}, which allows it to participate more fully in the network. On the other extreme, when x has c_{max} or more children, the portion of x's association trust value based solely on its base trust value can be no larger than b_{max}. The upper bound means that, no matter how many children x has, their actions—their base trust values—still have an effect on x's association trust value. Figures 2 and 3 show relatively the same results as those for Equation 2, except that the effect of the children's base trust values is dampened by the bounds. The lower bound b_{min} can be kept low at the beginning of the network in order to encourage peers to find

children. When the network matures, b_{min} can be raised to allow those peers without children because the market has been saturated to still participate meaningfully in the network.

If we look closely at Equation 3 in Figure 4, we see that a problem still remains. There is no penalty when x invites more than c_{max} peers to join the network. This situation is of little concern at first glance; however, it poses a serious problem. When a peer has a large number of children, the average of their base trust values has a dampening effect. The result is that when one of those multitude of children misbehaves, the effect on x's association trust value will be negligible. As a result, care must be taken in the setting of c_{max} to strike a balance between encouraging the growth of the network and keeping the effects of badly-behaving peers noticeable. Clearly, to have this effect, c_{max} must be enforced.

4.1.4 Equations 4 and 5

Equations 4 and 5 impose a penalty on x when it exceeds c_{max} children. Instead of summing the base trust values of all the children, the equations instead use the c_{max} lowest peers for the calculation. The result can clearly be seen in Figure 4, where a discernable dip in the association trust value of x is shown for both Equations 4 and 5. Note that a similar dip is not seen in Figures 2 and 3 because all the children have the same base trust value. If we assume that a peer tries to maximize its benefit in the network by maximizing its association trust value and that it is unlikely that the peer's children all have the same base trust value then we can conclude that our approach to discouraging extra children works as predicted. However, our analysis shows a small problem. The deterrent effect appears to much less than we would have expected. The penalty for adding peers in excess of c_{max} appears to be much less than the benefit of adding peers before reaching c_{max}. We would expect that the absolute slope of the line after c_{max} would be greater than or equal to the absolute slope of the line before c_{max}. We plan to address this problem in future work.

Equation 4, like all the equations previous to it, assumes that a single child needs to be invited in order for a peer to have the full benefit of its base trust value. Equation 5 introduces the concept of a minimum number of children, c_{min}, before this happens. At the start of a network, c_{min} can be set high in order to encourage the growth of the network. When the network matures, c_{min} can be lowered to allow those with only a small number of children to participate fully in the network.

4.2 Simulation

We test the validity of our approach using a simple round-by-round simulator written in Perl. We use the simulator to test our approach and compare it against EigenTrust, which we also use as the base trust mechanism. The results, while not conclusive, are very encouraging. We present a short description of the simulator and how it works followed by a discussion of the results that we obtained from it.

4.2.1 Simulator

We simulate the peer-to-peer network on a round-by-round rather than a simulation time basis. Each round consisted of possible new peers entering the system, transactions taking place, a recalculation of all peers trust values, and the dropping of peers if appropriate. We list the parameters for the simulation in Table 1 and discuss them below.

We used nearly the same size and composition of network as the original EigenTrust paper Kamvar *et al.* (2003). Their experiments were run with 60 good peers and 42 malicious peers. We ran our experiments with 60 good peers, 30 malicious peers, and 10 selfish peers. We define each peer type as simply as possible. A *good peer* sends a good file as a transacting offering peer and reports truthfully on what it received as a querying peer. A *malicious peer* is actually a traitor, as defined earlier, which allows it to build up some initial trust. As a transacting offering peer, it sends good files for the first 30 rounds after entering the network and then sends bad files thereafter. As a querying peer, it lies about the success of transactions, so that successful transactions, such as those with good peers and initially well-behaved malicious peers, are reported as unsuccessful (0) and unsuccessful transactions, such as those with badly-behaved malicious peers, are reported as successful (1). In the results section, we also look at how both our approach and

Parameter	Setting
Number of nodes in network	100
Number of good (G) nodes	60
Number of malicious (M) nodes	30
Number of selfish (S) nodes	10
Rounds that malicious peers act as good	30
Number of initial top-level peers	3
Number of rounds	1000
Rounds before maturity	200
Rounds before maturity with added peers	80
Percentage of peers querying per round	50%
Percentage of peers offering per query	20%
Minimum children (c_{min})	2
Maximum children (c_{max})	5
Lower bound on inviter's trust (b_{min})	0.25
Upper bound on inviter's trust (b_{max})	0.75
Grace rounds	30
Grace transactions	20
Grace threshold	85%
Rounds peer can go without offering (passive drop)	30

Table 1: Simulation settings.

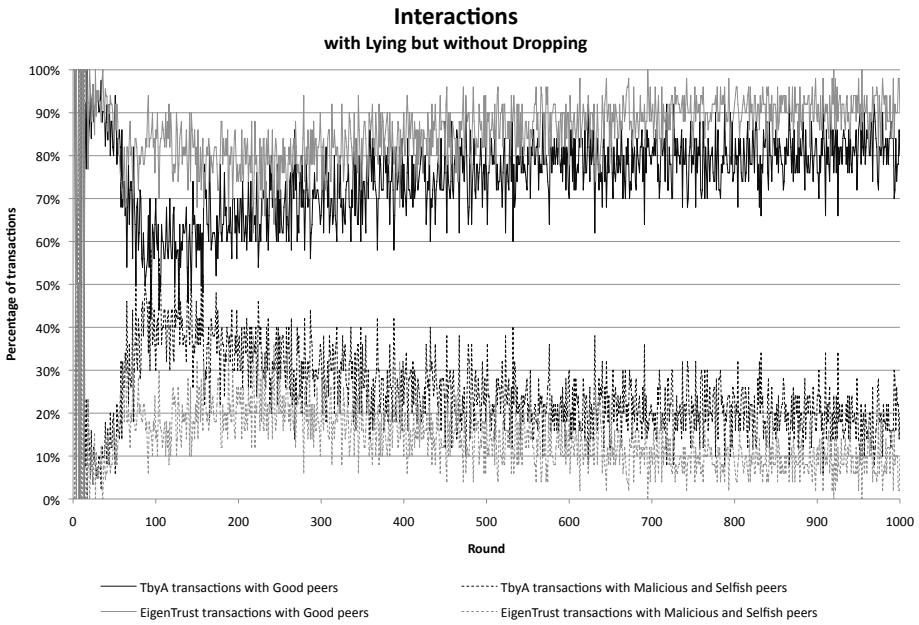

Figure 5: Interactions with good and malicious peers for both trust by association and EigenTrust when malicious peers lie and no dropping is allowed.

Drop Threshold	With lying		Without lying	
	Good	Malicious	Good	Malicious
25%	60	30	60	9
30%	60	29	60	16
35%	60	27	60	13
40%	60	16	60	19
45%	60	16	60	13
50%	60	15	60	7
55%	60	15	58	8
60%	48	15	54	10

Table 2: Number of surviving peers for various drop thresholds based on the percentage of reported successful transactions for networks where malicious peers lie or do not lie.

EigenTrust work when malicious peers do not lie. A *selfish peer* never offers anything and so can never be chosen as a transacting offering peer. As a querying peer, however, it reports truthfully on what it received.

The simulation starts with 3 good peers that are seeded with arbitrary trust values in order to act as the known trusted peers for the EigenTrust algorithm. These peers also act as the roots of the network branches for trust by association. The simulation is run over 1000 rounds with all the peers entering the system by round 200. Out of the first 200 rounds, 80 rounds are chosen randomly for the addition of new peers and up to 8 new peers are added in any one of those rounds until the network is fully populated. These rounds and the number of peers added during them are precalculated before the start of the simulation. In the runs shown in the results section, all the peers are added within the first 40 to 60 rounds.

Each round works as follows. The round starts with adding new peers, if any. The new peers are randomly assigned as invitees, up to 2 an inviter, to peers that have not reached their maximum number of children ($c_{min} = 2, c_{max} = 5$). Next, we want to encourage a large number of transactions, so 50 percent of the peers in the network are chosen randomly as querying peers. For each querying peer, 20 percent of all peers are chosen randomly as offering peers. Each querying peer chooses an offering peer 90 percent of the time based probabilistically on the offering peers' association trust values (or base trust values if we are testing EigenTrust). The other 10 percent of the time, the querying peer chooses an offering peer uniformly at random from those peers whose trust value is 0. Depending on the types of peers that are interacting, the transaction is recorded as successful (G/G, M/M, S/G) or unsuccessful (G/M, M/G, S/M) in the transaction table, where x/y denotes the *querying peer*/*transacting offering peer* combination. After the transactions are complete, the base trust values are recalculated for all the peers and then the association trust values are recalculated using Equation 5 ($b_{min} = 0.25, b_{max} = 0.75$). Finally, any peers that meet either the active or passive dropping criteria are dropped. We discuss the dropping criteria in the results section. There are a number of grace parameters that exclude peers from meeting the dropping criteria. No node can be dropped before it has either been in the network for more than 30 rounds or participated in 20 transactions. No peer with a trust value that is greater than 85% of the maximum association trust value can be dropped.

4.2.2 Results

In order to validate our results, we look at how trust by association (TbyA) compares with EigenTrust. Specifically, we want to see a reduction in the number of transactions with malicious peers. To examine this property, we look at the percentage of transactions that are with good peers as transacting offering peers (G/G, M/G, S/G) versus the percentage of transactions that are with malicious peers as transacting offering peers (G/M, M/M, S/M) for each round of the algorithm. We will refer to these as good transactions and malicious transactions, respectively. We take these two complementary values (they add to 100) and plot them for a run each of the two algorithms. The two runs use the same random seed and so have the same distribution of added peers.

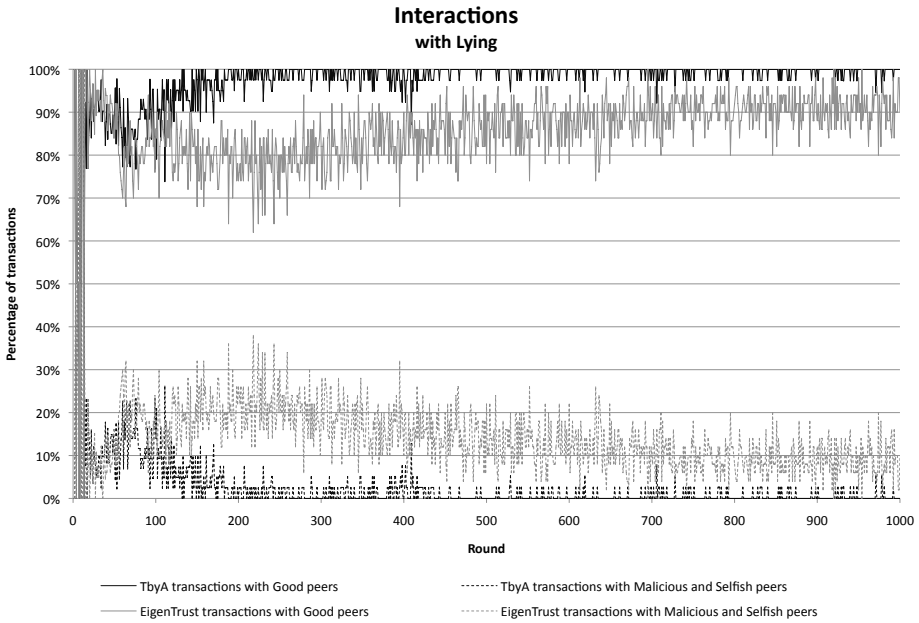

Figure 6: Interactions with good and malicious peers for both trust by association and EigenTrust when malicious peers lie and dropping is allowed with a drop threshold of 55%.

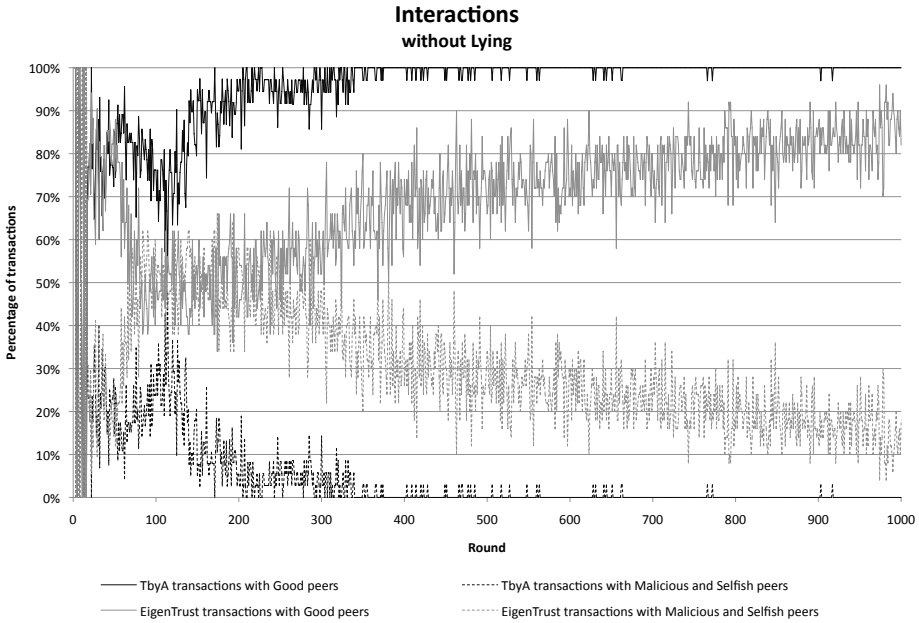

Figure 7: Interactions with good and malicious peers for both trust by association and EigenTrust when malicious peers do not lie and dropping is allowed with a drop threshold of 50%.

We first want to see if trust by association alone is comparable to EigenTrust. For the TbyA experiment, we allow malicious peers to lie but we do not allow the dropping of peers. The results of the two experiments are shown in Figure 5. All peers are added by round 45 in both experiments, so we are really focussing on the network at maturity and how each algorithm handles malicious peers. Trust by association is comparable to EigenTrust but performs worse in allowing roughly 10% more malicious transactions per round. The result is not unexpected given that offering peers are chosen probabilistically to participate in transactions using their association trust value. In our simulation, a peer's association trust value is made up of a random mix of good, malicious, and selfish peers. In the absence of dropping, good children of a malicious parent act as a damper against the negative influence of the parent's base trust value on its own association trust value. The result is an association trust value that is artificially high and a greater likelihood of being picked for a transaction.

We have discussed dropping before as a responsibility of the inviter. We want to give negative feedback to the inviter when she invites a malicious peer into the network, so that she can either apply social pressure to the peer to behave well or recommend the removal of the peer as a last resort. By allowing the inviters to take care of the possibly numerous single malicious peers, the administrators are free to go after the more dangerous malicious collectives. Our simulation, by construction, focuses on the single malicious peer problem. Obviously, the social aspect of trust by association is difficult to simulate. For our simulation, we wanted to develop a simple drop rule that would mirror the sort of advice given in on-line forums on "How to find the friend that is costing you money." It had to be something simple that could be presented as information to the inviter about her invitees and on which she could make an informed decision.

We implemented two kinds of dropping: passive and active. Passive dropping is simple. If a peer does not participate in the network by becoming an offering peer within 30 rounds, it is automatically dropped. The parameter is tuned to avoid removing good peers who are probabilistically starved of transactions. The effect in our simulation is that all selfish peers are removed 30 rounds after entering the network. In a real-world peer-to-peer network, the time between transactions would have to be much longer, on the order of days or weeks, to avoid removing a peer that has gone on vacation or something similar.

Developing an active drop rule turned out to be harder than we expected. Our first instinct was to look at the base trust values of a peer's children. Unfortunately, we could not consistently identify malicious peers using their base trust values alone. Nor could we identify them by other means including comparing their base trust values against the average for other children of the same peer, and looking at the slope of the linear regression of their base trust values over various periods of time. When we look at the trust values of a parent and its children, the malicious node is often easily identifiable, but not always in the same way. This variation is a result of the trust values being bounded and, therefore, normalized. All transactions in the system have a global effect on everyone's trust value that makes finding a relatively static point of reference difficult. We suspect that the problem is not intractable and it is possible to develop a simple indicator that can be used to both increase the cost in the system to peers with malicious children and help those peers identify the children causing the increase. Developing this indicator will be the subject of future work. For our simulation, we wanted to focus on the effect of removing the malicious peers from the system and we eventually found a simple rule that would allow us to do just that.

In lieu of a trust based active drop rule, we decided to look at the unnormalized data to which we had access. We looked specifically at EigenTrust's unnormalized transaction matrix, which contains a count of all the reported successful transactions. We say *reported successful* because we know that malicious peers lie about the success of their transactions. We realized that if we could compare the number of reported successful transactions with the total number of transactions, successful and unsuccessful, that that would be good indicator of whether a node was malicious or not. Our active drop criterion is whether a node falls below a certain threshold percentage for the *percentage of reported successful transactions*.

The number of surviving malicious nodes for various active drop thresholds are shown in Table 2. When malicious peers are allowed to lie, our active drop criterion eliminates half the malicious peers in the network when the threshold is set to 55%. The threshold may seem low except that we know that 30% of the network is malicious peers who lie and 10% of the network are selfish peers who never offer. So, 30 out of 90 transactions with good peers as transacting offering peers will be reported as unsuccessful meaning that our

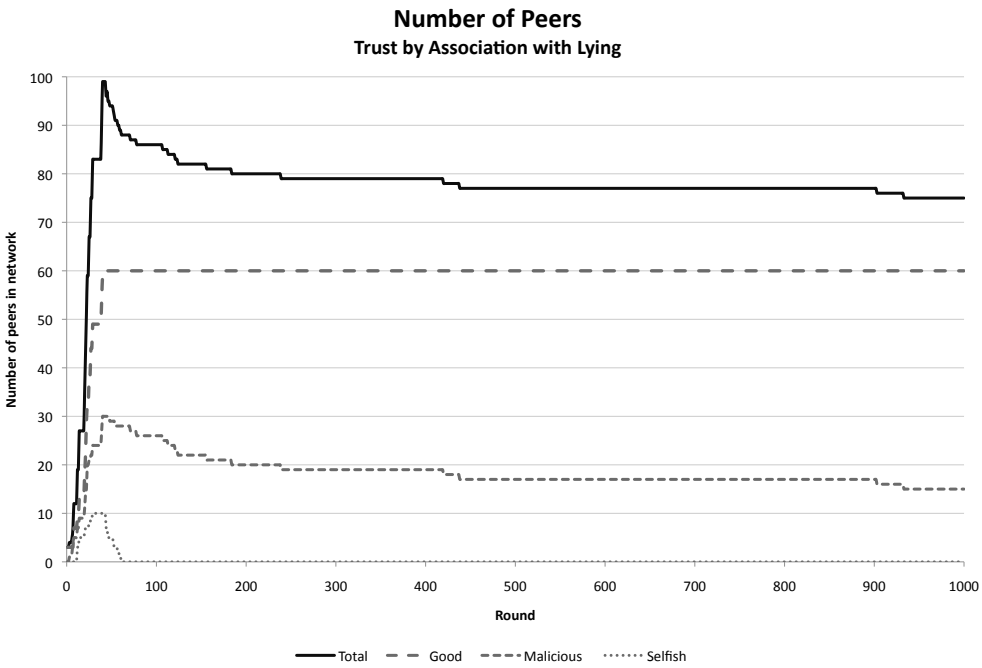

Figure 8: Number of peers in network for trust by association experiment shown in Figure 6 when malicious peers lie and dropping is allowed.

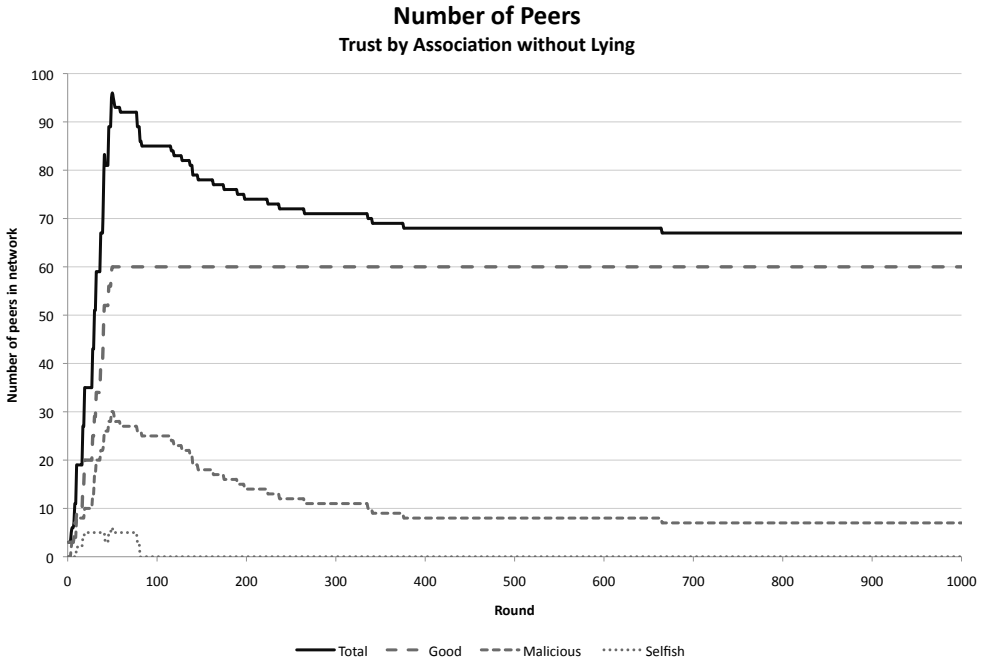

Figure 9: Number of peers in network for trust by association experiment shown in Figure 7 when malicious peers do not lie and dropping is allowed.

drop threshold has to be at least less than $1 - 30/90 = 0.667 = 66.7\%$ to avoid eliminating good peers. In fact, we found some good peers were eliminated when we set the threshold to 60%. Given our active drop criterion, it is likely that these good peers had too many transactions with malicious peers who lied about the transaction's outcome.

Figure 6 shows the comparison of EigenTrust versus trust by association with lying and a drop threshold of 55%. Trust by association does much better than EigenTrust with our new drop rule. Within 200 rounds, practically 100% of the trust by association transactions are with good peers, while EigenTrust drops to roughly 10%. EigenTrust is clearly doing its job in marginalizing malicious peers but it cannot marginalize them entirely. By eliminating even some of the malicious peers, we greatly limit the effect of the remaining peers in the network.

One question that might arise from these results is why not simply add dropping to EigenTrust? This approach ignores both the social aspect of trust by association, where social pressure can be used to improve behaviour in the network overall, and the ability to assign responsibility, which is especially useful when dealing with malicious collectives. In a normal peer-to-peer network, it may be possible to root out a malicious collective and remove them from the system, but in an invitation-only network we can go one step further to find out how the collective entered the network in the first place and, if appropriate, remove the well-behaving front peer or peers.

Our active drop rule is not one that could be implemented in a real world network. Because malicious peers lie, the percentage of reported successful transactions inverts when the number of malicious peers is at 50% or above. The result is that good peers will have less reported successful transactions than malicious peers. In order to target malicious peers, we would have to drop peers that had a percentage of reported successful transactions *above* a certain threshold. However, the rule is effective enough to show the benefits of dropping malicious peers.

We wondered what would happen if malicious peers did not lie. The results are both predictable and surprising. As might be expected, trust by association did better in finding and dropping malicious peers when they didn't lie as shown in Table 2. Figure 7 shows how trust by association with a drop threshold of 50% fared against EigenTrust. For the trust by association experiment, it took longer at around 300 rounds for the number of malicious transactions to approach 0 as opposed to around 200 rounds in the lying case; however, when it reach 0, there was less variation. The surprising result is how badly EigenTrust did when malicious peers told the truth. The EigenTrust result is reflected in the extra time it took for the trust by association experiment to converge near 0 malicious transactions. However, without the ability to drop malicious peers, the EigenTrust experiment took much longer to converge.

Figures 8 and 9 show the number of peers for each round of the trust by association experiment for the malicious peers' lying and not lying, respectively. The passive drop rule eliminates the selfish peers almost immediately. The active drop rule leads to malicious peers being dropped throughout the simulation.

5 Conclusion and future work

We believe that this work is the first to attempt to augment existing reputation trust mechanisms for P2P networks rather than trying to replace them. Our simple simulation shows that although trust by association performs strictly worse when only the equation is applied, it performs well when dropping is added. Our analysis of the equations show that they should have the desired effect if, as our work on the simulation showed, we can find a static point of reference from which to both identify malicious peers and give negative feedback to peers that invite them into the system. More importantly, our approach goes beyond the purely technical approaches of existing algorithms to include a social aspect. The invitation-only tree structure also gives us the ability to do more than address the malicious nodes in the network, but also to deal with, if appropriate, the peer or peers that invited them to join the network.

We have extensively discussed the nature of trust in P2P networks and how existing trust algorithms approach their calculations. In light of this discussion, we have proposed a simple but powerful approach to combining the trust values from existing trust algorithms to produce a result that is greater than the sum of its parts. Using the external relationships that must exist in the creation of an invitation-only P2P

network, we have created a mechanism that can encourage participants in the network to both take care in who they invite to join the network and to actively seek out and remove badly-behaving peers once they have joined. Trust by association has the potential to turn the typically "wild west," lawless environment of P2P networks into the equivalent of small towns where everyone knows everyone and all are concerned about their reputations. We believe the new environment is well suited to supporting e-commerce in a way that has so far eluded existing P2P networks.

Future work includes looking for a more robust way of identifying malicious nodes. We would also like to look at areas that we did not address with this simulation. We focussed here on how the algorithms reacted after the network had matured. It would be nice to see how they compare when new peers are added throughout the duration of the simulation. It might also be beneficial to add the concept of resources into our simulation. In the current simulation, a peer that is chosen randomly to offer is assumed to have the resource that was queried. In this case, an offering peer with a high trust value is more likely to get the transaction and thus increase its trust value even more. We would like to see how the system works when there are peers that have some resources that are relatively rare. In addition to a more realistic simulation, we would like to look at how trust by association interacts with other base trust models. Unfortunately, it is not possible to test the social aspects of our approach in anything short of a full-blown deployment. If further research shows that our approach has promise, hopefully one day it will be deployed.

References

Abrams, Zoë, McGrew, Robert, & Plotkin, Serge. 2005. A non-manipulable trust system based on eigentrust. *SIGecom Exch.*, **5**(4), 21–30.

Androutsellis-Theotokis, Stephanos, Spinellis, Diomidis, & Vlachos, Vasileios. 2007. The MoR-Trust distributed trust management system: Design and simulation results. *Electronic Notes in Theoretical Computer Science*, **179**, 3–15.

Barr, Joe. 2000 (May 12). *Opinion: Napster, Gnutella, and Internet guerrillas.* CNN.com. http://archives.cnn.com/2000/TECH/computing/05/12/mp3.guerrillas.idg/index.html (Accessed August 21, 2008).

Bharambe, Ashwin R., Herley, Cormac, & Padmanabhan, Venkata N. 2006. Analyzing and improving a BitTorrent network's performance mechanisms. *In: INFOCOM 2006. Proceedings of the 25th IEEE International Conference on Computer Communications.*

Breban, Silvia, & Vassileva, Julita. 2002. Using inter-agent trust relationships for efficient coalition formation. *Pages 221–236 of: AI '02: Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence.* Lecture Notes in Computer Science, vol. 2338. London, UK: Springer-Verlag.

Carter, Jonathan, Ghorbani, Ali A., & Marsh, Stephen. 2002. Architectural components of information-sharing societies. *Computational Intelligence*, **18**(4), 638–655.

Jones, Steve, & Marsh, Steve. 1997. Human-computer-human interaction: Trust in CSCW. *SIGCHI Bulletin*, **29**(3), 36–40.

Jonker, Catholijn M., & Treur, Jan. 1999. Formal analysis of models for the dynamics of trust based on experiences. *Pages 221–231 of: MAAMAW '99: Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World.* Lecture Notes in Computer Science, vol. 1647. Springer Berlin/Heidelberg.

Jun, Seung, & Ahamad, Mustaque. 2005. Incentives in BitTorrent induce free riding. *Pages 116–121 of: P2PECON '05: Proceedings of the 2005 ACM SIGCOMM Workshop on the Economics of Peer-to-Peer Systems.* New York, NY, USA: ACM.

Kamvar, Sepandar D., Schlosser, Mario T., & Garcia-Molina, Hector. 2003. The Eigentrust algorithm for reputation management in P2P networks. *Pages 640–651 of: WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM.

Kerr, Reid, & Cohen, Robin. 2006a (October). Guaranteed security in trust and reputation systems. *In: PST '06: Proceedings of the Conference on Privacy, Security and Trust*.

Kerr, Reid, & Cohen, Robin. 2006b (October). Modeling trust using transactional, numerical units. *In: PST 2006: Proceedings of the International Conference on Privacy, Security and Trust*.

Marti, Sergio, & Garcia-Molina, Hector. 2004. Limited reputation sharing in P2P systems. *Pages 91–101 of: EC '04: Proceedings of the 5th ACM conference on Electronic commerce*. New York, NY, USA: ACM.

Marti, Sergio, & Garcia-Molina, Hector. 2006. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, **50**(4), 472–484.

Papaioannou, Thanasis G., & Stamoulis, George D. 2006. Reputation-based policies that provide the right incentives in peer-to-peer environments. *Computer Networks*, **50**(4), 563–578.

Ranganathan, Kavitha, Ripeanu, Matei, Sarin, Ankur, & Foster, Ian. 2004. Incentive mechanisms for large collaborative resource sharing. *In: CCGrid 2004: Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*.

Ravichandran, Ajay, & Yoon, Jongpil. 2006. Trust management with delegation in grouped peer-to-peer communities. *Pages 71–80 of: SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM.

Ries, Sebastian. 2007. CertainTrust: A trust model for users and agents. *Pages 1599–1604 of: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*. New York, NY, USA: ACM.

Sabater, Jordi, & Sierra, Carles. 2001. REGRET : A reputation model in gregarious societies. *Pages 61–69 of: Papers from the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*. ACM.

Singh, Aameek, & Liu, Ling. 2003 (September). TrustMe: Anonymous management of trust relationships in decentralized P2P systems. *Pages 142–149 of: P2P 2003. Proceedings of the Third International Conference on Peer-to-Peer Computing*.

Thommes, Richard, & Coates, Mark. 2005. BitTorrent fairness: Analysis and improvements. *In: Proceedings of the Workshop on Internet, Telecommunications, and Signal Processing*.

Tran, Thomas. 2005. A reliability modelling based strategy to avoid infinite harm from dishonest sellers in electronic marketplaces. *Journal of Business and Technology*, **1**(1), 69–76.

Yu, Bin, & Singh, Munindar P. 2000. A social mechanism of reputation management in electronic communities. *Pages 154–165 of: CIA 2000: Proceedings of the 4th International Workshop on the Future of Information Agents in Cyberspace (Cooperative Information Agents IV)*. Lecture Notes in Computer Science, vol. 1860. Springer Berlin/Heidelberg.

Yu, Bin, Singh, M.P., & Sycara, K. 2004 (August). Developing trust in large-scale peer-to-peer systems. *Pages 1–10 of: MASS 04: Proceedings of the IEEE First Symposium on Multi-Agent Security and Survivability*.

List of Figures