Managing Reputation and Trust in Peer-to-Peer Networks.

CP4022 Research Topics in Networks and Distributed Systems.
Assessment 1

By Stacey Greenaway

Contents

1. Introduction

A peer-to-peer network is decentralized, where all nodes in the network act as both clients and servers and is powered by the bandwidth of all peers with ad hoc connections to the network.

This review will concentrate on filesharing instead of other p2p networks such as ecommerce or instant messaging, firstly because I am interested in the area and fortunately, the majority of research concerns itself with filesharing. All of the research papers I will review refer to Gnutella, the most popular application on the open source Gnutella network is bearshare. Other p2p networks independent of Gnutella are Soulseek, Bittorrent, kazaa etc there are too many to list.

The eBay feedback system is an example of a Trust and Reputation system currently in use. Buyers and sellers rate each other after each transaction and this feedback is shared with all users. A percentage reputation score is allocated to every user based on their feedback. Negative feedback has more weight than positive and is clearly indicated. The authors of each research paper reviewed reference the Ebay feedback system as an example of a reputation system.

Each of the four research papers reference research by Despotovic [1] as having relevance to their own research. He created a binary trust model which measured trust based on a successful interaction (1) or an unsuccessful interaction (0). This method has been developed by the researchers I have reviewed. Three of the researchers [3], [4] and [5] also cite Damiani et al [2] as having produced research that gave grounding for their research to develop. I have chosen to review only Damiani et al's research as it is more recent and proposes a protocol influential on the 3 other papers.

1.1 Definitions

Certain definitions need to be made which will be used throughout this review.

Trust - A peer's trust in other peers based on his own past experience.

Reputation - A peer's trust in another peer based on the experiences of other peers.

Therefore, trust models measure an individual peer's opinion of another peers reliability and honesty based on each individual's experiences at sharing files; Reputation Management systems share this trust measurement between all peers in order to gain an overall value of reputation for each peer.

Probability mathematics is useful when using simulations to replicate the real world as a way of replicating user behaviour. A probability equation gives peers a way to act that mimics a real world system. The following methods are used in two of the research papers reviewed.

Bayesian network - A Bayesian Network is a graph consisting of nodes and arcs. Nodes represent variables and the arcs represent the relationships and dependencies between the variables. [29] Used by Wang et al. [3]

Eigenvectors – A special set of vectors associated with Linear Algebra, and matrixes, where left eigenvector is a row of the matrix and right eigenvector is a column of the matrix. [30] [31] Used by Kamvar et al. [5]

The following terms are used throughout to describe the main participants of a file sharing peer to peer network.

File Provider – a peer providing a file for download

Servent – a peer who is both client and server.

Free Rider - A peer who only downloads and does not share any files. There are two possible reasons for free riders described by Damiani et al. a desire to avoid costs in terms of bandwidth and system load or a fear of being accountable for sharing illegally duplicated copyrighted material [2]. I would also propose a third one, which is a lack of trust in other peers and the application and maybe a lack of technical understanding of the system meaning, a peer may not want to open up their hard drive to unknown users.

Inauthentic files – Not all inauthentic files are malicious but are of no use to the peer who downloaded them. Some examples include viruses, (such as gnutella worm), files that say they are one thing but turn out to be another (eg says it is an artist's new single but is actually an old track by that artist), unreadable file type e.g. rmj and the user does not have real player, corrupted files and incomplete files.

1.2 Attacks and Threats

I will refer to threats on peer to peer networks that create the requirement for a trust and reputation system throughout the review, these are defined below.

Decoy files
A malicious peer will respond to any query with a copy of the requested file, but will deliver a file that has been tampered with or contains a virus a the point of download. [4] [5]

Malicious peer
A peer who either belongs to one of the groups below or will provide an inauthentic file for every request. [2] [3] [4] [5]

Malicious collective
A group of malicious peers who know each other and collaborate to subvert a P2P system. [3] [4] [5]

Malicious spies
Work in pairs, peer A pretends to be reputable by always providing authentic files, it will also give positive feedback to it's partner, peer B who will always provide inauthentic files. In some cases, peer A will respond positively to a file query, but peer B will actually provide the file at the point of download. [5]

Self Replication
The anonymity of networks means that a malicious peer can respond positively to all queries and return inauthentic content. Honest peers could share a malicious file they had downloaded without realising it is malicious. Certain viruses or worms such as gnutells.vbs worm can appear in the system as a peer then rename a copy of themselves to respond to a query and are downloaded. [2] [5]

Man in the middle
Malicious peer D intercepts the query between two honest peers, A and B and modifies the message so that their inauthentic file is downloaded when A thinks it is downloading an authentic file off peer B. [2]

Pseudospoofing
Most P2P systems use pseudonyms rather than full authentication to identify peers, malicious peers can therefore control multiple identities relatively easily. They can keep using a new pseudonym once they gain a bad reputation. The other false pseudonyms are used to give good reputation to other pseudonyms controlled by the one malicious peer spoofing the system. [2] [3] [4] [5]

ID Stealth
This is a variant of pseudospoofing and reduces the effectiveness of any reputation system. Upon receiving a query the malicious peer responds with 2 or more responses each response will appear to come from a different peer as only one response will contain the actual pseudonym of the malicious peer, the others will contain pseudonyms that the malicious peer has stolen from reputable peers in order to try and trick a peer into downloading a file from what it thinks is a reputable peer. [2]

Shilling
This is a well known problem facing auction sites where a malicious auctioneer will push up the price of his auction by having multiple registration names (shills) which can also be applied to P2P reputation systems in that a shill can be used to push up the reputation of other malicious peers. It is different from pseudospoofing as real IP addresses are used for each pseudonym (shill). [2]

2. Research Review

2.1 A Reputation Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks

(Damiani et al.) [2]

2.1.1 Introduction

Damiani et al propose a system called "XRep", a "self regulating system where the P2p Network is used to implement a robust reputation mechanism" [2]. Reputation is shared through a distributed polling algorithm, where peers can request the reliability of the resource offered by another peer before downloading it; reducing malicious content and eventually blocking it. Their approach they claim can be easily "piggybacked" on to existing p2p protocols such as Gnutella. Both the servent and the resource are given a reputation score. They do not discuss in any detail their experiments, except to say they used a modified version of an open source Gnutella client. [2]

2.1.2 XRep Protocol

The basic principal is that a peer, p, queries the network for other peer's opinions (votes) on resources and servents. Each peer maintains information on its experience with other peers and files in two repositories, a resource repository which records an ID for each file downloaded and whether it is good(+) or bad(-) and a servent repository storing the number of successful and unsuccessful downloads by each peer. The values in these repositories are used as "votes" and converted to binary, (utilising Despotovic's ideas [1]) where a positive (+) = 1 and negative (-) = 0.

XRep has six phases, Resource Searching, Resource Selection, Vote Polling, Vote Evaluation, Best Servent Check and Resource Downloading. Resource Searching and Resource Selection form the process of querying the p2p application and retrieving a result of files relevant to the query, then selecting one to download. With Xrep, instead of selecting the first file in a queue or selecting a file a random, trust and reputation values are considered before deciding on a file to download. The next phase therefore is Vote Polling, peer (p) asks the other peers opinions (poll request) about the resource (r) it is about to download or on the servent (s) offering the resource. To protect integrity and confidentiality each poll request includes a public key with which the poll responses are encrypted. Upon receiving a request each servent checks its repositories and sends a message back to the peer with the vote encrypted in the key called "pkpoll" and also their IP Address and Port. The fourth phase is vote evaluation, p needs to evaluate its collection of votes on the resources and their servents to determine their reliability, this process should highlight any malicious activity. Firstly the votes are decrypted to detect and discard any that have been tampered with, thus combating any man in the middle attacks. Secondly, to detect malicious attacks such as Pseudospoofing, p clusters the votes, which allows it to detect those sharing the same IP address. An average value of all votes in the cluster is calculated and returned to the querying peer (p). Finally, a random selection of "voters" from each cluster is contacted for confirmation of their vote using the IP and Port encypted in pkpoll. The authors state that this process can combat Shilling attacks by forcing "the attacker to pay the cost of using real IPs as false witnesses" [2], however I cannot see that there is any procedure in place that stops the malicious peer from lying again and confirming a positive vote for an inauthentic resource or malicious servent. If not enough peers respond positively in a given time frame the random selection process is repeated. At the end of this phase, the querying peer knows which servents and resourses are trustworthy. The fifth phase, Best Servent Check is the process of determining which reputable servent to download the authentic resource

from. The chosen servent will be the one with the highest reputation value. This can cause a bottleneck as every peer will chose that servent to download from, hence increasing his reputation more until he dominates the system. Damiani et al do not propose any strategy to over come this other than a final security measure to check to see if the chosen servent actually has a copy of the authentic resource by again messaging the servent. If the servent has a copy of the resource then download it, other wise query the next reputable peer on the list. This procedure is beneficial in reducing ID stealth attacks. The sixth and final phase of the Xrep protocol is for p to update his repositories with his opinion of both the servent and resource. [2]

2.1.3 Analysis

The XRep protocol combines the reputations of both the resource and the servent which is beneficial if a resource is new or only going to be used once, as it can adopt the reputation of its servent. Equally, new users joining the system can take advantage of a resources reputation and quickly improve their reputation score by sharing trusted resources. This, the authors claim can overcome a problem of reputation systems, *cold start*, where new peers struggle to participate in the system because of a lack of reputation. Which leads into the major problem with reputation systems, performance bottlenecks, where all downloads are directed to the most reputable peer, at each download his reputation increases more, so he will eventually command a majority share of reputation for a specific resource. Eventually the download speed associated with this servent will be so slow under the high demand no one will be able to receive the resource from him. He will therefore start to receive negative votes reducing his reputation and reducing the bottleneck, but not solving the problem as once his reputation begins to rise again the same situation will occur, this process will shift the bottleneck onto the next reputable peer, not eradicate it.

The authors compare resource based and servent based reputation systems, based on certain considerations. Firstly a reputations life cycle is considered, resource based reputations it is suggested will have a wider scope and longer life cycle than that of servent based reputation as a good resource will always be recognisable as such, regardless of who offers it. Secondly, the impact on a peer's anonymity is considered, it is suggested that a servent based reputation system does not compromise anonymity as reputation is linked to the pseudonym, it therefore relies on pseudonyms being kept persistent. A resource's reputation is not linked to the pseudonym so the peer can take a fresh identity at every interaction. Thirdly, it is highlighted that resource based reputation does not support *blacklisting* as it is the peer's IP address that would be used to blacklist and no link is made between the resource and the IP address at which it is stored. A fourth aspect considered is data storage and bandwidth requirements of any reputation system. The authors state that as the amount of resources outnumber the amount of servents, any resource based system will require more storage and bandwidth to store and calculate the reputations of all the resources in the system, than storing the reputations of all the servents. [2]

2.1.4 Conclusions

This protocol appears to be a convoluted process of messaging that can only result in overloading the system with query and response messages. I doubt that present bandwidth limitations could cope with the level of messaging involved with this proposed system. There is also a lack of consideration for usability I cannot see that any user would want to wait the amount of time all this messaging would take before downloading a file. Also, peers may not be sat at their computer while they are sharing files so will not be available to respond to the messages immediately. It has occurred to me that some messaging would be handled by the application and not involve user interaction, but not all of it. Filesharing is something that happens in the background, this system doesn't account for this. Despite these drawbacks, the XRep protocol describes a basic reputation system for choosing reputable peers and authentic files which can be extended into a more efficient system with further development, which the following three research papers have all attempted.

2.2 Trust and Reputation Model in Peer-to-Peer Networks

(Wang et al.) [3]

2.2.1 Introduction

Wang et al's research evolves concepts of trust and reputation from the very basic trustworthy – not trustworthy differentials made by Despotovic[1] and Damiani[2] to a more versatile interpretation of trust. They begin by suggesting that trust and reputation depend on certain contexts, that it is multi faceted meaning that multiple aspects determine whether or not a peer is trustworthy and that it is dynamic in that trust and reputation can increase or decrease over time. An analogy to explain this could be that Mike has two friends John who is a mechanic and Bob who is a Doctor. Mike trusts Bob with a medical complaint but not to fix his car and respectively, trusts John to fix his car but not to diagnose a medical condition. So in the context of fixing a car John is trustworthy, but Bob is untrustworthy. They propose a Trust and Reputation model for peer-to-peer networks that uses a strategy of probalistic maths called a Bayesian Network to build a profile of each peer's opinions based on different contexts of trust. [3]

2.2.2 The Trust and Reputation Model

The model builds two kinds of trust that measure a peer's reliability, firstly the trust that "peer A has in peer B's capability of providing services" [3] and secondly the trust "that peer A has in peer B's reliability in providing recommendations about other peers" [3]. The authors measure reliability with two factors; "Truthfulness", whether a peer is truthful when delivering its opinions of another peer and "Similarity", whether one peer has similar preferences and ways of judging a file sharing interaction as another peer. [3]

As peers are never exactly the same, they have different requirements of a system. What one peer may consider a good file is not what another peer would consider good. For instance peer A's priority in a good file is its content regardless of its quality; whereas peer B may only consider a file a good file if it is of the highest quality. An example could be that peer A considers a file good if it is M4a format as it is compatible with her ipod which is where she will play the file, peer B on the other hand may consider this file bad as he has to convert it to mp3 in order to play the file on his PC. In this case peer b cannot trust peer A's recommendation of the file. It is important for a peer to ask for recommendations from peers they trust implicitly rather than asking a large random selection of peers as this saves time and computational cost as well as providing the peer with a good set of recommendations.

After querying the system the peer receives a list of files and file providers, they can sort the list in order of trust in the file provider. The peer chooses a file provider from the top of the list and downloads the file. If the peer is unsure of their trust in the servent they can request recommendations from other peers. Referees give recommendations to a peer, if the peer agrees with the referee's recommendation then trust in the referee increases, otherwise it decreases. Fig 1 below illustrates this process.

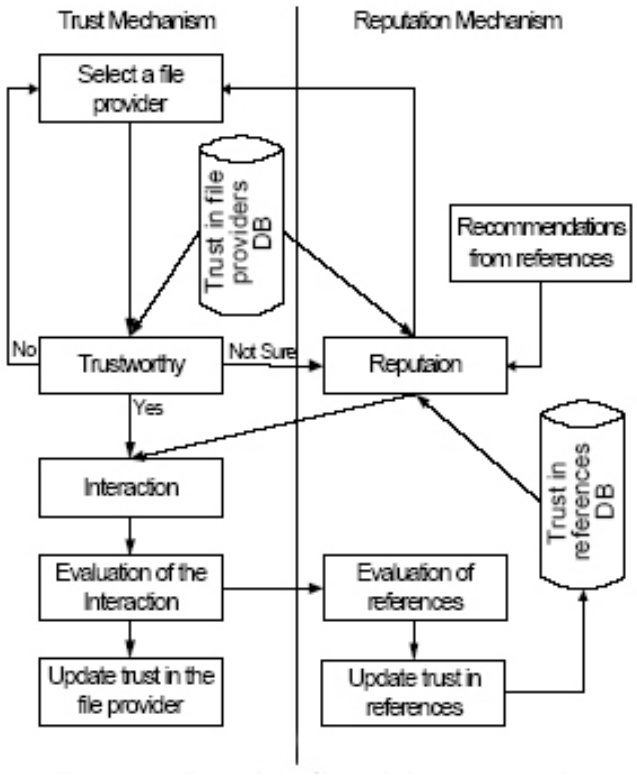

Fig 1 "Functionality of the trust and reputation mechanism on board of the peer" [3]

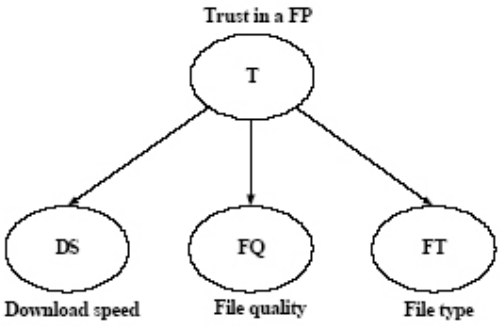

Fig 2 "A Bayesian network model" [3]

They continue to discuss how a peers needs may differ depending on the situation. Sometimes a peer may want a general overview of a file provider's capability and sometimes they may only be concerned with download speed. On another occasion the peer may require a file of good quality and at a fast download speed. The peer needs to base its trust in another peer based on two capabilities, it needs to compare its trust in the file providing peer based on file quality and download speed.

To try and solve this problem Wang et al use a naïve Bayesian Network, meaning a basic Bayesian network consisting only of one parent and several child nodes. Their BN Model is illustrated above in Fig 2. FP is the file provider, T represents a peers trust in the file provider, which is measured as a percentage of all positive interactions. The diagram shows that this trust is dependant on certain factors, in

the case of this diagram, Download Speed (DS), File Quality (FQ) and File Type (FT). More user preferences can be added to the Bayesian Network, for instance they might want to ascertain if the file is free from copyright. As trust is dependant on all or some of the conditions in the network, trust only need be calculated once regardless of a change in conditions increasing its efficiency.

Similarly to Damiani et al's protocol [2], every interaction with a file provider is evaluated by a peer as satisfying or unsatisfying and their Bayesian Networks are updated accordingly. Wang et al's description of their system for exchanging recommendations holds more similarities to Damiani et al's proposed protocol, the difference being that the Bayesian Network is queried instead of stored repositories. In Wang et al's system peer A requests recommendations about a file they wish to download. They use the example that the peer wants to download a movie of high quality at a fast download speed and requests recommendations on the peers providing the desired file from various referees. Upon receipt of this request referees will check their Bayesian Networks and return whether they found the file provider to meet these requirements. Peer A receives the responses which could be from trusted peers, malicious peers or unknowns. The authors state that responses from untrusted peers will be discarded, there has however been no mention of how a peer would know if they are untrusted, if only recommendations based on the file are requested. They continue to discuss how the recommendations of the trusted peers and unknowns are combined to get the total recommendation, but again there is no explanation of how the peer knows it can trust the unknown referee's opinion. They assume that the peer has a group of peers it knows it can trust, based on past experience, the recommendations from this group of peers is weighted more heavily than the unknown peers recommendations as they share similar preferences. The total recommendation value is the value of recommendations from trusted peers combined with the value from unknown peers, this is compared to a threshold value, if the total recommendation is greater than the threshold then the file will be downloaded otherwise not. After a successful download the Bayesian Networks of the file provider and the referees will be updated to reflect the peers trust in them. The authors represent this process as a formula (fig 3 below).

$$tr_{ij}^{n} = \alpha * tr_{ij}^{o} + (1-\alpha) * \varepsilon_{\alpha}$$

Fig 3 [3]

tr_{ij}^{n} denotes the "new trust value" that peer i has in peer j; [3]

tr_{ij}^{o} denotes the "old trust value"; [3]

α denotes the "learning rate" and is a number between 0 and 1; [3]

ε_{α} denotes the "new evidence value", if the recommendation value is greater than the threshold value then the new evidence value is returned as 1 otherwise -1 is returned. [3]

Another way the authors discuss to determine whether a peer is reliable as a referee is building a comparison between the Bayesian Networks of file providers. Peers can exchange and compare their BN's to enable them to find other peers sharing similar preferences and develop a network of trusted peers to use as referees. After each comparison trust in each peer is updated according to the formula illustrated below in fig 4.

$$tr_{ij}^{n} = \beta * tr_{ij}^{o} + (1-\beta) * e_{\beta}$$

Fig 4 [3]

β denotes the "learning rate" and is a number between 0 and 1;

e_{β} denotes the "new evidence value" and is a number between -1 and 1.

The Bayesian Network contains a peers opinions on all past interactions with a specific file provider, therefore comparing two BN's is the same as comparing all past interactions of the two peers which is considerably more efficient and informative. Whereas the formula in fig 3 calculates peer trust based on one interaction, the formula in fig 4 calculates peer trust based on all past interactions, meaning that the trust value calculated carries more weight than that of fig 3. Therefore the value of $\beta > \alpha$.

In order to compare Bayesian Networks it is assumed by the Authors in their simulations that all peers have the same structure of BN. This is potentially unrealistic in a real world scenario, and could have implications on the success of this system on a real P2P network. Wang et al describe the comparison process as,
"Peer 1 gets the degree of similarity between the two Bayesian networks by computing the similarity of each pair of nodes (T, DS, FQ and FT), … and then combining the similarity results of each pair of nodes with different weight in order to take into account peers' preferences."[3]
Via this process a peer can create a group of trusted peers he knows share similar preferences regarding files. The recommendations of these peers will be weighted higher when calculating the recommendation value from a selection of referees.

The author's have assumed during their discussion of their recommendation system that all peers are "truthful in their recommendations". [3] They suggest though that by using this recommendation system if a peer gives a false recommendation on another file provider in order to try and boost another malicious peers reputation i.e. in pseudospoofing, shilling or malicious collective attacks, then this will be highlighted as it will differ greatly from the recommendations of the trusted peers. The recommendation can either be discarded or at least it will be counteracted by the weight of the trusted peers recommendation values. The peer will then give the malicious peer a negative reputation score when updating their BN and the malicious peer's reputation will decrease, reducing their power in the system.

2.2.3 Experiments

Wang et al developed a simulation of a file sharing system in a P2P network. They state that "for simplicity each node in our system plays only one role at a time, either the role of file provider or the role of a peer." And that "each peer only knows the peers directly connected to it and a few file providers". [3]
They further describe their system as,
"Every peer has an interest vector. The interest vector is composed of five elements: *music, movie, image, document and software*. The value of each element indicates the strength of the peer's interests in the corresponding file type. The files the peer wants to download are generated based on its interest vector. Every peer keeps two lists. One is the peer list that records all the
other peers that the peer has interacted with and its trust values in these peers. The other is the file provider list that records the known file providers and the corresponding Bayesian networks representing the peer's trusts in these file providers. Each file provider has a capability vector showing its capabilities in

different aspects, i.e. providing files with different types, qualities and download speeds." [3]

Each of ten runs of the experiment involves interactions between 10 file providers and 40 peers. Peers compare their Bayesian Networks after every 5 interactions. There are 1000 interactions in total. Each run is evaluated by taking averages of the results. [3]

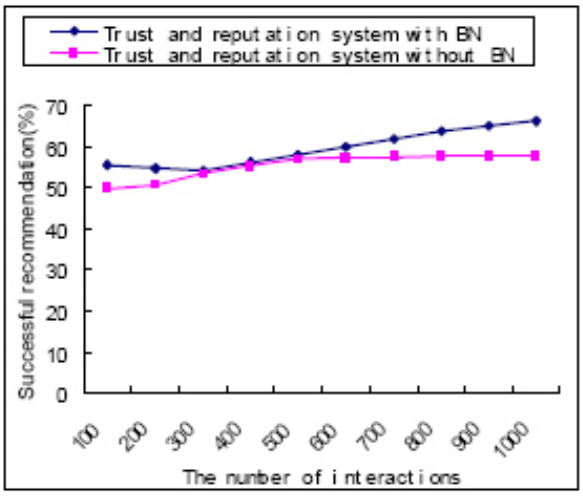

Fig 5 Experiment 1 [3]

The first experiment Wang et al. conducted was fairly basic to prove that a trust and Reputation system using their Bayesian Network Trust Model would perform better than one that didn't use a BN. The % successful recommendation is calculated by dividing the number of successful recommendations by the number of positive recommendations. This eliminates the results where a negative recommendation was received as this would not result in an interaction with the file provider. Fig 5 above illustrates their results and shows that the system using a BN trust model performs slightly better.

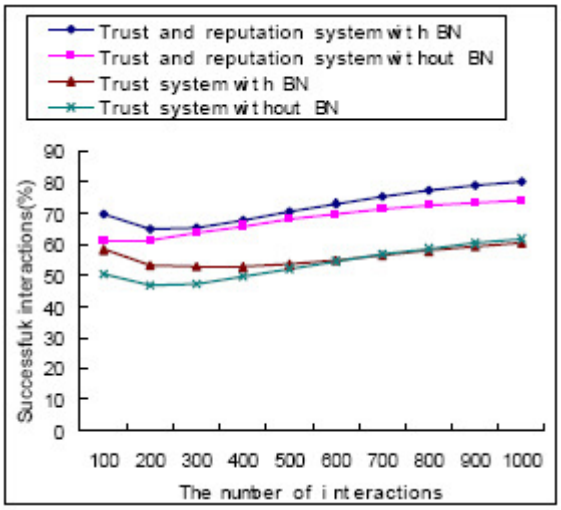

Fig 6 Experiment 2 [3]

The second experiment Wang et al. conducted aimed to compare 4 different systems, they used the two systems from experiment 1, a "Trust and reputation system with BN" and a "Trust and reputation system without BN" and a "Trust

system with BN" and a "Trust system without BN". The two latter systems do not allow peers to "exchange recommendations with each other" all trust is gathered through their own experience. [3] The results are illustrated in Fig 6 above and reflect the results of experiment 1 that a system using their Bayesian Network Trust Model performs moderately better than the other systems. It also shows an anomaly between the results of a Trust system with and without BN, as the system without a BN actually performs slightly better than the system with a BN with a higher number of interactions. The authors explain this result as that they have used "an imprecise BN due to insufficient experience". [3] These results do highlight that the success of any trust and reputation system comes from having peers exchange their opinions of other peers and their resources.

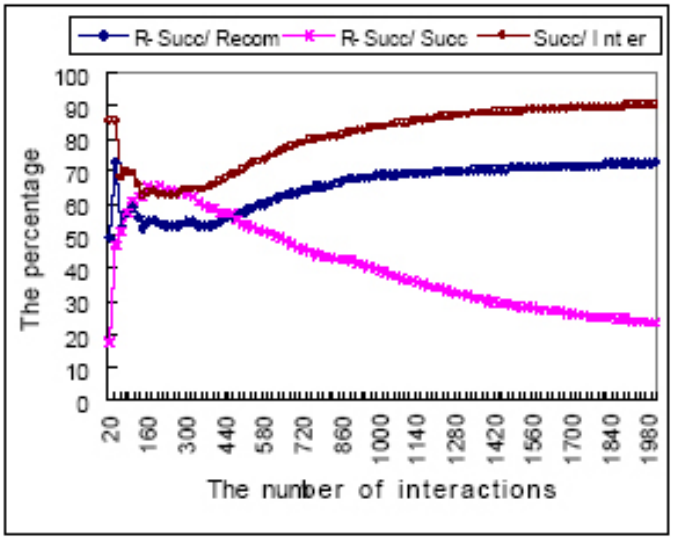

Fig 7 Experiment 3 [3]

For the third experiment Wang et al increased the number of interactions between peers and file providers to 2000 and used a trust and Reputation system with Bayesian Trust Model. They measured the following parameters:

"*R-Succ/Recom*, the percentage of successful recommendations, which is the number of successful recommendations divided by the number of positive recommendations.

R-Succ/Succ, the percentage of successful interactions based on recommendations, which is the number of successful interactions based on recommendations over the number of all successful interactions.

Succ/Inter, the percentage of successful interactions in all interactions, which is the number of successful interactions divided by the total number of interactions." [3]

The results illustrated in Fig 7 surprisingly show that the amount of successful interactions is greater when recommendations are not taken into account, indicating that peers will download a file anyway and take a chance maybe for speed in a real world situation, and more often than not they will receive an authentic file as the majority of file providers are trustworthy. The authors state R-Succ/Succ to be the most interesting result as it shows that with more interactions peers need less and less recommendations, as they gain more experience with file providers and develop their Bayesian Networks they have a better idea of which peers in the network they can trust. [3]

2.2.4 Conclusions

The authors conclude their experiments with discussion of future developments for their system which includes more focus on performance, giving two examples of research "how fast a peer can locate a trustworthy service provider and how fast the workload of file providers can be balanced." They also pose a question "How many BN's can a peer afford to maintain?" [3] The summary of which seems to be that only a small amount of BN's can be stored making the system suitable only for small networks, being optimal in a situation where two peers may wish to trade files between one another. This is extremely un realistic in a real world P2P network and also drastically limits the choice of resources available to a peer.

There is no discussion by the authors of how they propose to make their system secure and how it can avoid attacks such as pseudospoofing and Shilling. As there is no encryption described with the Bayesian Network model, it seems that they could easily be intercepted and modified by a malicious peer. The system seems particularly vulnerable to malicious collectives, pseudospoofing and shilling attacks as they can easily manipulate each others BN's and increase reputation values.

They do discuss the major problem of performance bottlenecks and suggest that their system has a solution to it in that all peers will download from a file provider with high trust until eventually his download speed will decrease (performance bottleneck), therefore his reputation will decrease easing the bottleneck as the peers will download from the next file provider with high trust. This is not a solution as it only shifts the performance bottleneck on to another peer.

This system offers possibilities in that it considers the complicated nature of trust, that it cannot be solved by a simple honest/dishonest equation. However like Damiani's research [2] there is no indication to the amount of processing speed and memory and therefore time that is needed to query and exchange each individual Bayesian Network, which will impact on the overall usability of the P2P file sharing application.

2.3 A Reputation-Based Trust Management System for P2P Networks
(Selcuk et al.) [4]

2.3.1 Introduction

Selcuk et al. propose a protocol to control the amount of inauthentic files a malicious peer can disperse through the P2P network. The proposed system will differentiate between malicious responses to a query and trustworthy responses by querying "trust Vectors" which are kept locally by peers; the querying peer can consult its own "trust vector", or request a "trust rating" from other peers in the system. [4] In comparison to previously reviewed research, this system uses a simulation of a Gnutella system, and uses messages to gain recommendations from other peers.

2.3.2 Measuring Trust

Trust Vectors are binary and typically consist of 8, 16, or 32 bits, their length is stored as an integer variable. They contain opinions about all of a peers past transactions. After each download a peer updates their own trust vector with either a positive (1) or negative (0) opinion about the other peer. Each positive or negative opinion is represented in the vector as 1 bit. So these trust vectors have the potential to record an extensive amount of information about a peers download history. The result is recorded at the vectors most significant bit, being the first bit in the vector, all previous values are shifted right. Fig 8 below depicts this process. It shows that peer A's trust vector stores the history of 4 interactions at the start of the download, 3 positive and 1 negative, after an authentic file is downloaded from peer B, peer A's trust vector is updated with 1 bit representing the positive interaction.

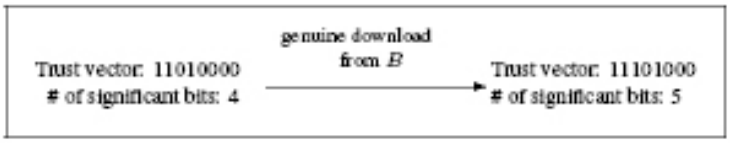

Fig 8 [4]

A Trust Rating is calculated by dividing the sum of the Trust Vector by the power of 2, then dividing the result by 2 to the power of the number of significant bits in the vector. A distrust rating is calculated with the same calculation; both of these calculations are portrayed in Fig 9 below.

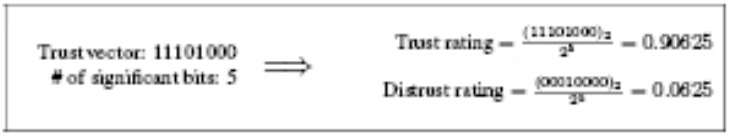

Fig 9 [4]

The authors explain the importance of distrust ratings as follows, "handling the distrust ratings separately has the additional feature of not letting a dishonest dealing be erased easily by a few honest transactions, which closely models real-life trust relations where a single dishonest transaction in someone's history is a more significant indicator than several honest transactions." [4]

After issuing a query and having had a list of file providers returned, the trust ratings of these peers need to be evaluated. This is done by taking an average of the trust values of the most trusted peers in the list; the amount of peers to use in the calculation is determined by a pre set threshold value. If the number of

trusted peers in the list is less than the threshold, a trust query is issued to a random selection of peers in the network to obtain enough trust values to perform the calculation. Responses to this query include both the trust and distrust ratings for a file. Selcuk et al. propose a credibility rating as a way of giving weight to the opinions of the peers who respond to the request. The rating is calculated from a credibility vector, which is similar to a trust vector except that it stores a 1 when a peer's opinion of a file is truthful and 0 when a peer's opinion is considered untruthful. As in the trust rating evaluation, a threshold value is established, setting the number of responses to evaluate. How weight is allocated to a trust rating is described by the authors as follows, "if peer A issued a trust query on peer B, and the responses of peers $R_1;R_2.... ;R_k$, $k \leq \theta_C$, qualify for consideration, and A's credibility rating for R_i is c_i and R_i's trust rating for B is t_i, then A's queried trust score on B is…" (fig 10 below) [4]

$$\frac{\sum_{i=1}^{k} c_i t_i}{k}.$$

Fig 10 [4]

An average distrust rating can also be calculated with the same equation only using other peer's distrust ratings instead of their trust ratings.

Credibility vectors are not as straightforward to update as trust vectors. A positive (1) is only added to the vector if the querying peer deems the opinion of the responding peer to match his experience of the download. If the peer's experience of the download was negative, but the response from the peer was positive (and vice versa), the credibility vector will have a 0 added. Any distrust rating greater than zero counts as a negative opinion, which has more weight than any positive opinion. To use the eBay feedback system as an example, when looking at feedback you will take more account of any negative feedback and weight your decision of whether to or not a seller is trustworthy on the presence of negative feedback.

2.3.3 Security

Similarly to Damiani et al [2] who propose that a public key be attached to the query messages, Selcuk et al. propose a method of attaching a public key to a pseudonym, where the pseudonym would be an ID number of uniform length. This would authenticate the peer when their trust information is queried.

To test their system's performance under various malicious attacks, Selcuk et al. set up a series of simulations based on a Gnuttella client. The type of attacker considered were, naïve, hypocritical, malicious collective and Pseudospoofing. Naïve describes a malicious peer who will respond to every query with a malicious file, and is in a sense the control in the experiment. Hypocritical describes a malicious peer who for the most part responds with an authentic file, but will occasionally send a malicious file. Malicious Collective and Pseudospoofing are defined in section 2. Definitions, as they are types of attack referred to in other research.

Their simulation consisted of 1000 peers, of which between 1% and 10% were malicious, and 1000 files. Each peer held 10 files initially. Each peer was linked to three neighbours; a query would be submitted over these links for 3 hops, as specified by a Time To Live (TTL) for the query. File requests were issued at random, a peer would check to see if they had a copy of the file locally, if not, a query message was issued for the file and the author's protocol executed.

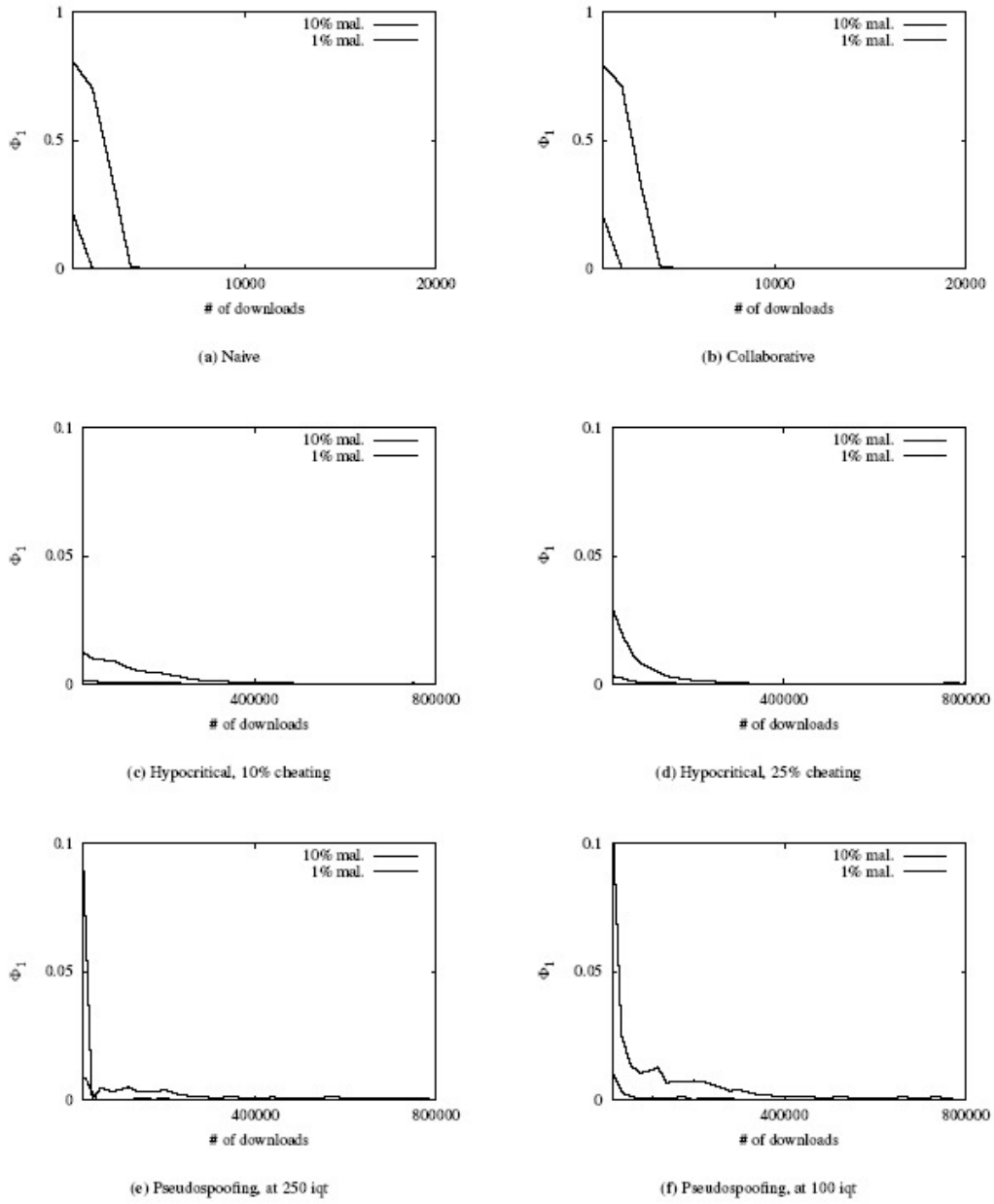

Fig 11 Results of experiments by Selcuk et al [4]

Φ_1 - represent's the ratio of malicious to all downloads. [4]

The results displayed in Fig 11, suggest that the authors protocol is adept at limiting the number of malicious downloads and within a short time.

They propose future development of this protocol to include provision for the hashes of malicious files to be stored and this information used to send out a warning message with any query that returns the files. Which although a good idea, could have an impact on performance of the system.

2.3.4 Conclusions

This paper is the first to propose giving values to distrust, which is a powerful aspect in identifying and eliminating malicious peers. The amount of honest transactions it will take for a malicious peer to counteract any dishonest transaction in order to gain credibility in the system will take up too many resources for it to be viable.

The authors make a statement which concerns me, "Once the file version to be downloaded is decided, the peer to download it from is selected randomly among those who offered that version, not considering the trust ratings. This way of selection has the desirable feature of enabling new peers to build a reputation as well as not overloading the trusted peers." [4] If no regard is given to trust ratings in the eventual download, what is the point of working out the trust ratings? It would make it possible for a malicious peer to be selected as file provider in the random selection. I can see the benefit of a random selection to avoid a performance bottleneck on the most reputable peer, but the file should be selected from a random selection of trusted file providers. This way it is unlikely that the same peer will be picked each time, avoiding any bottleneck.

By only providing trust and reputation to the file there is no room for consideration, as mentioned by Wang et al. [3], of the different preferences of peers. If a peer gives a file a negative trust rating because it took a long time to download as it is a large file of high quality, this would result in a distrust rating for the file, when to another peer whose main requirement is a files quality and download speed is not an issue, they are going to overlook this file thinking it to be malicious because there is no distinction as to why it got the negative rating. This questions the proposed credibility rating, as what one peer deems positive may not reflect another's opinion.

2.4 The EigenTrust Algorithm for Reputation Management in P2P Networks

(Kamvar et al.) [5]

2.4.1 Introduction

Kamvar et al. propose an algorithm for reputation management of Peer to Peer Networks. Their work is referenced by Selcuk et al. and shares a similar aim, "to decrease the number of inauthentic files in a P2P file sharing system that assigns each peer a unique global trust value based on the peers history of uploads". [5] The algorithm uses eigenvectors (defined in section 2. Definitions) as calculations to measure trust.

The emphasis of this research is on limiting attacks on P2P systems from malicious peers. They describe many threats including Decoy Files, Self Replicating Worms, Malicious Collectives, Malicious Spies and Pseudospoofing. All of theses are described in section 2. Definitions. They concentrate on identifying the malicious peer rather than the inauthentic files as the number of inauthentic files in a system can considerably outnumber the amount of malicious peers. [5] This is contrary to Selcuk et al's [4] research which proposed isolating the files.

Also proposed are a number of design considerations, namely, that anonymity is maintained by assigning reputation to a pseudonym rather than IP Address; that reputation is obtained through constant good behaviour, newcomers start with zero reputation and build upon it, removing the incentive for malicious peers to assume new identities once a bad reputation is earned and finally that minimal overhead will be enforced with relation to computation and message complexity. [5]

Kamvar et al. highlight a problem of previous research [2], [3], [4] of how to calculate, store and share trust values without a database and without clogging the system with messages requesting peers' trust values at every query. Their proposal promises to aggregate trust values keeping message complexity to a minimum. [5]

My knowledge of eigenvectors and linear algebra in general is minimal so I have relied heavily on the authors description of their algorithm in further discussion.

2.4.2 EigenTrust Algorithm

"Peer i is more likely to trust the opinions of peers from whom he has had an honest interaction with in the past." [5] This is the basis from which the EigenTrust algorithm has been developed. Values of *Local trust* and *Global Trust* are computed using eigenvectors. *Local Trust Value* is defined as S_{ij} and is calculated from peer *i*'s experiences downloading from other peer's, *j*. *Global Trust Value* is calculated from the local trust values assigned to peer *i* by peers *j* based on their experiences downloading from *i.* [5]

Part of the proposal to make their trust model more efficient than previous research is by normalizing the trust values. A criticism of this by Selcuk et al. is that by normalising they are loosing valuable information about the reputation of the peer. They give an example, if there are *n* identical trust ratings then their normalised value will be 1/n, regardless of whether the originals were the highest or lowest possible value. [4] Therefore peers are either honest or dishonest, there is no grey area or reputation scale. One honest peer can not be any more trustworthy than another honest peer. Alternatively a peer who makes one mistake by sharing an inauthentic file accidentally will be classed as malicious as a peer who has shared thousands of inauthentic files.

Kamvar et al. argue that without normalisation of trust values, malicious peers can assign arbitrarily high local trust values to other malicious peers and low local trust values to good peers, subverting the system. A normalised local trust value is defined as C_{ij} , the equation is depicted in Fig 12 below. This equation ensures that all values are between 0 and 1. [5]

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$$

Fig 12 [5]

Some drawbacks to normalising are mentioned: "normalised trust values do not distinguish between a peer, *j* that peer, *i,* did not interact with or had a bad experience with. Also C_{ij} values are relative and there is no absolute interpretation. If $C_{ij} = C_{ik}$ we know peer *j* has the same reputation as k in the eyes of peer I, but we do not know if it is good or bad reputation". [5] They say that despite these drawbacks they decided to normalize this way to avoid normalizing at every iteration, which is costly and that it leads to an elegant probalistic model. [5]

Similarly to research in [2], [3] and [4] peers share their opinions on other peers and interactions and their opinions weighted by the amount of trust placed in those peers by the querying peer. This process is expressed as an equation shown in Fig 13 below.

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

Fig 13 [5]

t_{ik} is the trust peer *i* places in peer *k* based on asking peers he knows he can trust the opinion of.

In matrix notation C is the matrix C_{ij} and $\vec{t_i}$ is the vector containing the values t_{ik}, so $\vec{t_i} = C^T \vec{c_i}$. This allows each peer to view the network that is wider than his own experience. However trust values still only contain values relating to *i* and the experiences of his friends. To get a wider view peer *i* will ask his friends' friends, fig 14 shows this equation. [5]

$$(t = (C^T)^2 c_i)$$

Fig 14 [5]

If peer *i* continues to ask the friends' of friends for their recommendations, he will eventually build up a view of the entire network. This is expressed in fig 15 below. [5]

$$(t = (C^T)^n c_i)$$

Fig 15 [5]

Kamvar et al. explain, "If n is larger the trust vector $\vec{t_i}$ will converge to the same vector for every peer *i*. It will converge to the left eigenvector" (row) "of C" (the matrix). " In other words $\vec{t}$ is a global trust vector in this model. Its element t_j qualifies how much trust the system as a whole places in peer *j*." [5]

Three peer groups are specified by the authors as having importance in the computation of trust. Each group is expressed as an equation. The first group is pre trusted peers, these can be defined as peers who are known to be trustworthy e.g. the creators of the network. A set of pre trusted peers is represented as P. The distribution of pre trusted peers throughout the system is represented as $\vec{p}$. Pre trusted peers are used to calculate trust values in the following situations. In the case of an inactive peer, a peer who doesn't download from anyone or assign trust values to other peers will have an undefined normalized trust value (C_{ij}). In this case the trust value of the pre trusted peers is used. This is defined by the equation represented in fig 16 below. [5]

$$c_{ij} = \begin{cases} \frac{max(s_{ij},0)}{\sum_j max(s_{ij})} & \text{if } \sum_j max(s_{ij},0) \neq 0; \\ p_j & \text{otherwise} \end{cases}$$

Fig 16 [5]

The trust value of pre trusted peers is also used when malicious collectives are highlighted in the system. By forcing all peers to have some trust in pre trusted peers they cannot get stuck in a collective receiving falsified trust ratings. The equation for this is depicted in fig 17 below.

$$\vec{t}^{(k+1)} = (1-a)C^T\vec{t}^{(k)} + a\vec{p}$$

Fig 17 [5]

All these equations joined together gives the basic EigenTrust algorithm, represented in fig 18 below.

$\vec{t}^{(0)} = \vec{p};$
repeat
  $\vec{t}^{(k+1)} = C^T\vec{t}^{(k)};$
  $\vec{t}^{(k+1)} = (1-a)\vec{t}^{(k+1)} + a\vec{p};$
  $\delta = ||\vec{t}^{(k+1)} - \vec{t}^{(k)}||;$
until $\delta < \epsilon;$

Fig 18 – Basic EigenTrust Algorithm [5]

In the basic EigenTrust algorithm each peer computes its own *Global Trust Value* and stores it locally. These values are then shared with other peers. Kamvar et al. suggest that this has implications on security of the system as malicious peers can report false trust values enabling them to appear in the system as honest peers and easily spread inauthentic files. To amend this, the authors suggest a *Score Manager* system, where *Score Managers* are peers who compute the trust value of another peer in the system. A peer has many *score managers*, so if a *score manager* happens to be a malicious peer who reports an untrue trust value, this will be counteracted by the results given by trusted peers.

To assign *Score Managers* to peers they use a Distributed Hash Table (DHT) which uses hash functions to map keys such as IP Address and TCP port into points in a logical coordinate space. This coordinate space is partitioned over the network so that every peer covers a region of that dynamic space. In this case, the key consists of a Unique ID for each peer which is made up of IP address and TCP Port, the peer who covers the region where that ID is hashed becomes that peers score manager. [5]

Fig 19 below illustrates this system, which is a modified version of CAN – Content-Addressable Network. (Ratnasamy et al. 2001 (13) cited by Kamvar et al. [5]) It depicts how peer *i*'s Unique ID has been mapped into points covered by peers 2,3 and 5 by the hash functions h1, h2 and h3, therefore these peers become the *Score Managers* for peer *i.* [5]

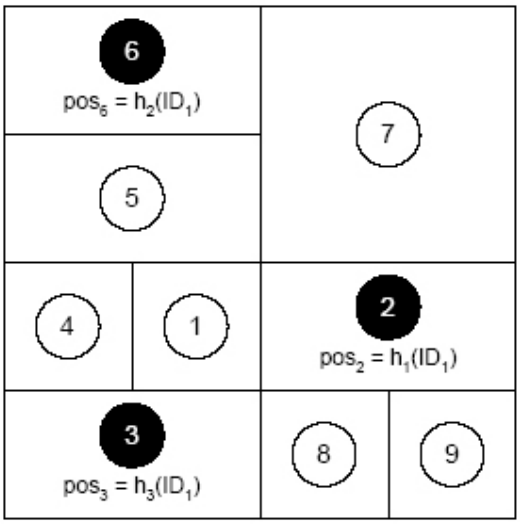

Fig 19 – Assigning *Score Managers* using DHT [5]

Since each peer also acts as a *Score Manager* it is assigned a set of Daughters Di, this set contains the peers for whom it is to compute the *Global Trust Values*. The *score manager* also maintains an opinion vector c_d^i which is sent out to peers who query it to find out the trustworthiness of the daughter peer, d, where d is in the score managers set of daughters ($d \in D_i$). If any of these peers download from d, then the score manager will learn the set of peers who did, A_d^i once they return their trust assessments as to the trustworthiness of d. The *score manager* will find out a set of peers who d has downloaded from B_d^i when d submits its trust assessments of those interactions. Fig 20 below depicts the full EigenTrust algorithm implementing the formulas described above. [5]

foreach *peer* i **do**
  Submit local trust values $\vec{c_i}$ to all score managers at positions $h_m(pos_i)$, $m = 1 \ldots M-1$;
  Collect local trust values $\vec{c_d}$ and sets of acquaintances B_d^i of daughter peers $d \in D_i$;
  Submit daughter d's local trust values c_{dj} to score managers $h_m(pos_d)$, $m = 1 \ldots M-1, \forall j \in B_d^i$;
  Collect acquaintances A_d^i of daughter peers;
  foreach *daughter peer* $d \in D_i$ **do**
    Query all peers $j \in A_d^i$ for $c_{jd} p_j$;
    repeat
      Compute $t_d^{(k+1)} = (1-a)(c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \ldots + c_{nd}t_n^{(k)}) + ap_d$;
      Send $c_{dj}t_d^{(k+1)}$ to all peers $j \in B_d^i$;
      Wait for all peers $j \in A_i^d$ to return $c_{jd}t_j^{(k+1)}$;
    until $|t_d^{(k+1)} - t_d^{(k)}| < \epsilon$;
  end
end

Fig 20 Secure EigenTrust Algorithm [5]

2.4.3 Analysis

Kamvar et al. discuss how their proposed algorithm increases system security. They suggest that anonymity is maintained as a peer's Unique ID is never revealed to its *score manager*. Therefore malicious peers cannot increase the reputation of other malicious peers, subverting the system. It also reduces the risk of Shilling or Pseudospoofing attacks as the malicious peer cannot copy the unique ID of one of its daughters in order to spread malicious content in the guise of an honest peer. Another proposed security benefit is that the peers are placed in the hash space at random; they cannot select which coordinates they want to be placed at, nor can they compute the hash value of their unique ID and place themselves at a position to be able to compute their own global trust value, again limiting the possibility of malicious attack.

The benefits of using *Global Trust Values* are suggested by the authors as being, firstly, "to isolate malicious peers from the network" and secondly, "to incent freeriders to share files by rewarding reputable peers". Malicious peers will be isolated from the system because peers will only download files from peers with high reputation. The authors suggest rewarding peers with high trust values by increasing their connectivity to other reputable peers, providing greater bandwidth, as a way of giving free riders an incentive to share files. [5]

2.4.4 Experiments

Kamvar et al. simulate a small Gnutella style P2P Network, the full settings can be seen in fig 21 below.

Network	# of good peers	60
	# of malicious peers	42
	# of pre-trusted peers	3
	# of initial neighbors of good peers	2
	# of initial neighbors of malicious peers	10
	# of initial neighbors of pre-trusted peers	10
	# Time-to-live for query messages	7
Content Distribution	# of distinct files at good peer i	file distribution in [16]
	set of content categories supported by good peer i	Zipf distribution over 20 content categories
	# of distinct files at good peer i in category j	uniform random distribution over peer i's total number of distinct files
	top % of queries for most popular categories and files malicious peers respond to	20%
	top % of queries for most popular categories and files pre-trusted peers respond to	5%
	% of time peer i is up and processing queries	uniform random distribution over [0%, 100%]
	% of time pre-trusted peer i is up and processing queries	1
	% of up-time good peer i issues queries	uniform random distribution over [0%, 50%]
	% of up-time pre-trusted peer i issues queries	1
Peer Behavior	% of download requests in which good peer i returns inauthentic file	5%
	% of download requests in which malicious peer i returns inauthentic file	0% (varied in Section 7.3)
	download source selection algorithm	probabilistic algorithm (varied in Section 7.2)
	probability that peer with global trust value 0 is selected as download source	10%
Simulation	# of simulation cycles in one experiment	30
	# of query cycles in one simulation cycle	50
	# of experiments over which results are averaged	5

Fig 21 Simulation Settings [5]

In order to simulate the randomness at which peers interact and share files they used Zipf Distribution of probability, where files were grouped into categories according to their content and popularity. Each experiment was run five times and

the results are then averaged for comparison. The authors state that their primary concern in analysing results is the amount of inauthentic and authentic files each node (peer) in the network uploads and downloads. They only simulate a small network, which the settings in fig 21 depict, but they state that their system will work on a larger scale, (but provide no evidence of this). Malicious peers are highly active in the system and respond to up to 20% of the queries issued. This does not imitate real world systems, which are in reality less hostile. Results are evaluated by comparison to a simulated P2P network without a Trust Model.[5]

As previously discussed one of the main problems with Trust and Reputation systems are performance bottlenecks. As is the case with all four research papers reviewed, it is a difficult problem to overcome. Kamvar et al. offer a solution to this. They propose a probalistic algorithm as a way of determining the download source, fig 22 below:

$$\frac{t_i}{\sum_{j=0}^{R} t_j}$$

Fig 22 [5]

They set a 10% probability of downloading from a peer who has a trust rating of 0, this it is suggested, gives new peers in the network the chance to be selected as a download source and to begin building a reputation. However this certainly also allows for a malicious peer to be selected as the download source, which discredits the purpose of obtaining trust values. The solution offered if a malicious peer is selected and an inauthentic file downloaded, is to delete the file and repeat the random selection process until an authentic file is downloaded. This seems to be a waste of resources. In a real world network, peers do not have unlimited resources (time and bandwidth) to download numerous files. It invalidates security procedure in that a virus could be downloaded. This also questions the usability and Quality of Service of the system, peers will not want to use a system where several versions of a file need to be downloaded before they get the file they requested. Fig 23 below depicts the results gained from running the probalistic algorithm to select download source. The results suggest that their proposal has some success, but does not perform as well as a random selection, that would happen in a p2p system without a trust model.

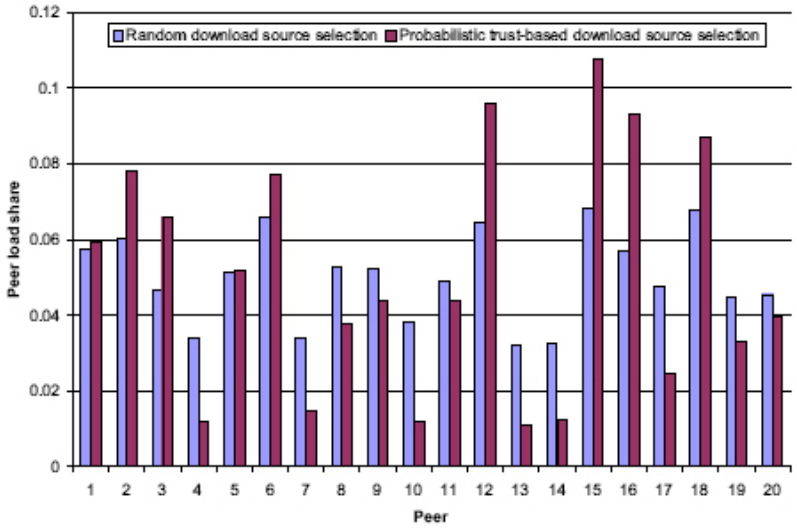

Fig 23 Combating Performance Bottleneck [5]

The main purpose for Kamvar et al's experiments is to test the proposed EigenTrust Algorithm against numerous Threat Models, namely individual malicious peers, malicious collectives and malicious spies. Definitions for these can be found in section 2. Definitions. In the first of the threat model experiments, Threat Model A, the effect of individual malicious peers, is analysed. The amount of malicious peers is increased by 10% in each experiment to a maximum of 70%. Fig 24 below shows the results:

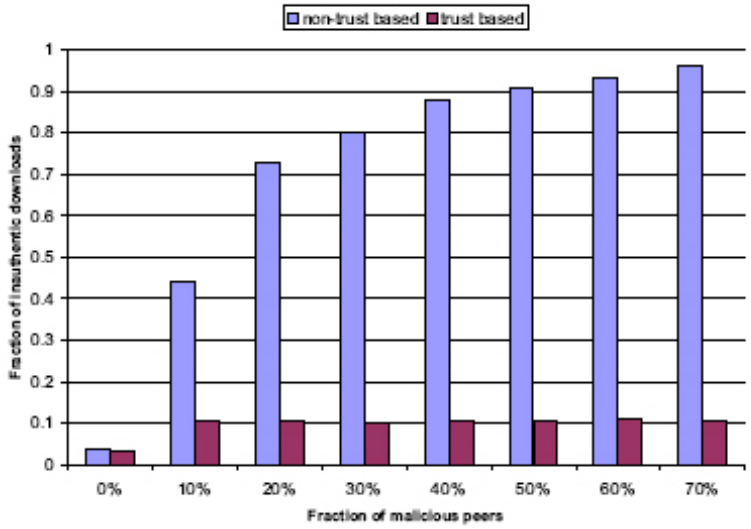

Fig 24 Threat Model A [5]

These results clearly show that the proposed system does reduce the number of malicious peers contributing inauthentic files to the system. Inauthentic files make up approximately 10% of the network compared to a maximum of over 90% in a network without the proposed trust model. The authors explain the presence of inauthentic files as the fact that some honest peers may not delete an inauthentic file they have downloaded and is inadvertently sharing the file. [5]

The second experiment, Threat Model B, tested the performance of the EigenTrust Algorithm against malicious collectives. Fig 25 below displays the results:

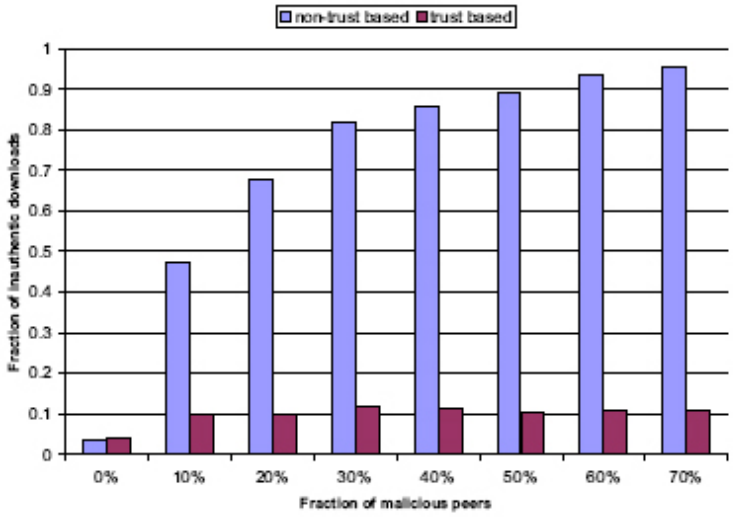

Fig 25 Threat Model B [5]

The results are very similar to the results of experiment 1 and the algorithm can be deemed as successful in limiting the amount of inauthentic files. The authors accredit this success to the section of the algorithm that uses the trust values of Pre Trusted peers to break up malicious collectives. [5]

The third experiment, Threat Model C tested the resilience of the system against malicious collectives with camouflage, which Selcuk et al. referred to in their research as hypocritical peers [4]. These are malicious peers who will offer authentic files some of the time in order to gain higher trust values, to increase their chance at being selected as a download source. Fig 26 below displays the results of this experiment:

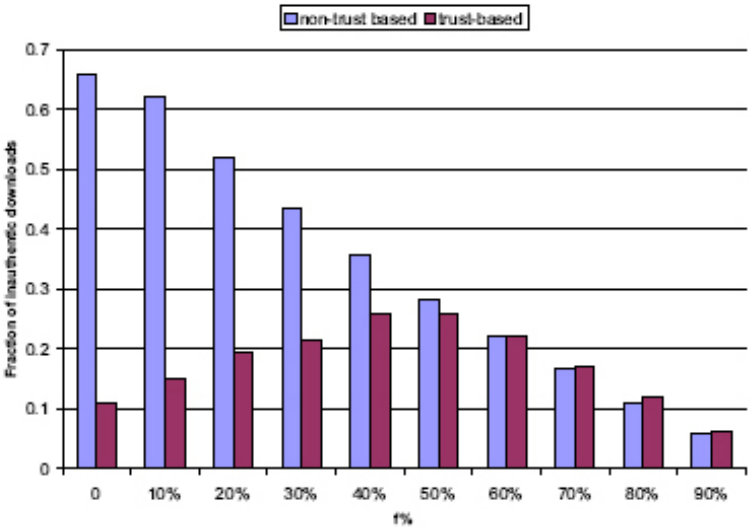

Fig 26 Threat Model C [5]

The results show that the more authentic files a malicious peer provides, the more impact they have at providing inauthentic files. Kamvar et al. endorse this result by the fact that to maintain this level of interaction is too costly in terms of bandwidth and disk space as the system forces the peer to provide more authentic files than inauthentic and all these files have to be stored and uploaded. Similar resources are required in the case of Threat Model D, malicious spies, in that the malicious spy needs to provide a large amount of authentic files in order to gain high enough reputation not be detected as a malicious peer so it can continue to give high trust ratings to another malicious peer who will provide the inauthentic files. The malicious peer providing the inauthentic files will be isolated from the system using the EigenTrust algorithm, therefore eradicating the threat. The authors discuss Pseudospoofing attacks as a threat, but they did not simulate this attack. They propose their system would combat any such attack by not assigning any value to new peers, thus removing the incentive for a malicious peer to create a new identity once they have earned a low global trust rating. [5]

2.4.5 Conclusions

In conclusion kamvar et al. discuss how their system reduces the number of inauthentic files in a network and how their eigenvector matrix of normalised local trust values takes into account the entire systems history for one single peer. They also say how by rewarding reputable peers with better quality of service is an incentive for honest peers to police their own file repository for inauthentic files and also to provide authentic files and not free ride on the system. However they only give suggestions on how to reward them, i.e. greater bandwidth, they do not suggest how this greater bandwidth will be provided. There are no results showing

if their DHT system is actually more efficient than passing messages between peers as the other researchers propose, [2], [3], and [4] to share recommendations. It would be interesting to see a comparison and to see how much strain the DHT system would put on a large P2P network and what effects this could have on bandwidth.

3. Conclusions

From the research presented in the four papers, I can see why no strategy has as yet been implemented on file sharing networks. Wang et al. [3] touch on how hard it is to measure trust and reputation when individual peers have different standards and expectations of files and download experience. There are no standards that define what trust and reputation is and how it should be measured.

None of the research proposes any sufficient solution to performance bottlenecking. Taking into account all suggestions to ease the bottleneck it could be suggested that by choosing a file to download from a random selection of the most reputable peers, where it is unlikely the same peer will be selected any more regularly than another peer, could avoid a performance bottleneck. Fig 23 in the review of Kamvar et al's work shows that random selection is the best solution to avoiding bottlenecks. There is also no consideration given to the idea that peers will be placed in a queue before downloading and that this place in the queue could be revealed in order to spread the traffic between reputable peers. This system is in operation by file sharing application SoulSeek.

Although file sharing P2P networks are not monitored and a lot of the content on there is illegal in terms of circulating copyrighted material, aspects of Quality of Service still apply, for instance a peer wants to be able to use the files he has downloaded, and wants them to be relevant to the criteria that he was searching for. However policing Quality of Service in P2P networks is virtually impossible due to their decentralized nature. Each individual peer is responsible for the quality of the content they provide only, this is why Trust and Reputation systems will be valuable to a P2P system, as they will offer some potential at increasing Quality of Service by reducing the amount of inauthentic files on the network.

I am concerned how much pressure these systems will put on a network, this doesn't seem to have been tested and they have only been simulated on small networks. The amount of resources needed to simultaneously query, download and upload files as well as querying and updating trust records is immense. This will require a lot of bandwidth, which is not infinite or free. Considering current bandwidth measurements, the proposed systems are likely to put too much strain on the connection. I don't think these proposed systems would work in a real world application yet, but they offer a lot of scope for development.

4. Bibliography / References

4.1 Reviewed research papers

[1] Aberer, K. Despotovic, Z. (2001) Managing trust in a peer-2-peer information system, *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM 2001), Atlanta,Georgia, November 2001*.

[2] Damiani, E. di Vimercati, D. C. Paraboschi, S. Samarati, P. Violante, F. (2002) Reputation-based approach for choosing reliable resources in peer-to-peer networks, *Proceedings of the 9th ACM Conference on Computer and Communications Security***.**

[3] Wang, Y. Vassileva, J. (2003) Trust and Reputation Model in Peer-to-Peer Networks, *Proceedings of IEEE Conference on P2P Computing, Linkoeping, Sweden.*

[4] Selcuk, A. A. Uzun, E. Pariente, M. R. (2004), A Reputation-Based Trust Management System for P2P Networks, *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, USA*.

[5] Kamvar, S. D. Schlosser, M. T. and Garcia-Molina. H. (2003) The eigentrust algorithm for reputation management in P2P networks, *Proceedings of the Twelfth International World Wide Web Conference***.**

4.2 References

[6] Abrams, Z. Mcgrew, R. Plotkin, S. (2004). Keeping peers honest in eigentrust. *In Workshop on Economics of Peer-to-Peer Systems.*

[7] Andrade, N. Mowbray, M. Lima, A. Wagner,G. Ripeanu, M. (2005) Influences on cooperation in BitTorrent communities *Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems.* pp111 – 115.

[8] Arthur, D. Panigrahy, R. (2006) Analyzing BitTorrent and Related Peer-to-Peer Networks. *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* Miami, Florida, pp961 - 969

[9] Cornelli, F. Damiani, E. De Capitani di Vimercati,S. Paraboschi, S. and P. Samarati. (2002) Choosing reputable servents in a P2P network. *Proceedings of the* Eleventh International World Wide Web Conference, Honolulu, Hawaii.

[10] Daswani, N. Garcia-Molina, H. (2002) Query-Flood DoS Attacks in Gnutella. [0nline] Stanford University [cited 5th April 2006]
<http://dbpubs.stanford.edu/pub/2002-26>

[11] Detsch, A. Gaspary, L. P. Barcellos, M. P. Cavalheiro, G. G. H. (2004) Towards a flexible security framework for peer-to-peer based grid computing. *Proceedings of the 2nd workshop on Middleware for grid computing*. 76 pp52-56.

[12] Dewan, P. Dasgupta, P. Pride: peer-to-peer reputation infrastructure for decentralized environments. *International World Wide Web Conference Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* pp480 - 481

[13] Dumitriu, D. Knightly, E. Kuzmanovic, A. Stoica, I. Zwaenepoel, W. (2006) Denial of Service Resilience in Peer to Peer File Sharing Systems. SIGMETRICS'05, June 6–10.

[14] Good, N. S. Krekelberg, A. Usability and privacy: a study of Kazaa P2P file-sharing *Proceedings of the SIGCHI conference on Human factors in computing system.* pp137 - 144.

[15] Guha, R. Kumar, R. Raghavan, P. Tomkins, A. (2004) Propagation of trust and distrust. *Proceedings of the 13th World Wide Web Conference*.

[16] Gupta, M. Judge, P. Ammar, M. (2003) A reputation system for peer-to-peer networks. *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, Monterey, CA, USA, pp144 – 152.

[17] Kamvar, S. D. Schlosser, M. T. and Garcia-Molina. H. (2003) Eigenrep: Reputation management in p2p networks. Unpublished work.

[18] Lee, J. (2003) An end-user perspective on file-sharing systems. *Communications of the ACM* Technical and social components of peer-to-peer computing. **46** (2) pp49-53

[19] Loo, A. L. (2003) The future of peer-to-peer computing. *Communications of the ACM, Why CS students need math*. 46 (9) pp 56 – 61.

[20] Marti, S. Garcia-Molina, H. (2004) Limited reputation sharing in P2P systems. *Proceedings the 5th ACM conference on Electronic commerce*.

[21] Mengshu, H. Xianliang, L. Chuan, Z. (2005) A trust model of p2p system based on confirmation theory. *ACM SIGOPS Operating Systems Review,* **39** (1), pp.56-62.

[22] Resnick, P. Zeckhauser, R. (2001) Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. Working Paper for the NBER workshop on empirical studies of electronic commerce.

[23] Rodriguez, P. Tan, S. Gkantsidis, C. (January 2006) On the feasibility of commercial, legal P2P content distribution. ACM SIGCOMM Computer Communication Review **36** (1) pp 75-78

[24] Singh, A. Liu, L. (2003) TrustMe: anonymous management of trust relationships in decentralized P2P systems. *Proceedings. Third International Conference on Peer-to-Peer Computing.* pp142- 149

[25] Stoica, I. Morris, R. Karger, D. Kaashoek, F. Balakrishnan, H. (2001) Chord: A scalable peer-to-peer lookup service for internet applications. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp.149-160.

[26] Xiong, L . Liu, L (2002) Building trust in decentralized peer-to-peer electronic communities Fifth International Conference on Electronic Commerce

[27] Xiong, L . Liu, L (2004) Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering.

[28] Ye, S. Makedon, F. Collaboration-aware peer-to-peer media streaming. *Proceedings of the 12th annual ACM international conference on Multimedia POSTER SESSION: Technical poster session 2: multimedia networking and system support.* pp412 – 415

4.3 URL's

[29] Wikipedia, the free encyclopedia (no date) Bayesian Network [online]. [cited 14th Apr 2006]. http://en.wikipedia.org/wiki/Bayesian_Network

[30] Wikipedia, the free encyclopedia (no date) Eigenvector [online]. [cited 14th Apr 2006]. http://en.wikipedia.org/wiki/Eigenvector

[31] Eric W. Weisstein. "Eigenvector." (no date) MathWorld--A Wolfram Web Resource. [cited 14th Apr 2006].
<http://mathworld.wolfram.com/Eigenvector.html>