A Trust based Access Control Framework for P2P File-Sharing Systems

Huu Tran Michael Hitchens Vijay Varadharajan Paul Watters
Department of Computing
Macquarie University
{huutran, michaelh, vijay, pwatters}@ics.mq.edu.au

Abstract

Peer-to-peer (P2P) file sharing systems have become popular as a new paradigm for information exchange. However, the decentralized and anonymous characteristics of P2P environments make the task of controlling access to sharing information more difficult, which cannot be done by traditional access control methods. In this paper, we identify access control requirements in such environments and propose a trust based access control framework for P2P file-sharing systems. The framework integrates aspects of trust and recommendation models, fairness based participation schemes and access control schemes, and applies them to P2P file-sharing systems. We believe that the proposed scheme is realistic and argue that our approach preserves P2P decentralized structure and peers' autonomy property whist enabling collaboration between peers.

1. Introduction

Peer-to-Peer (P2P) is driving a major paradigm shift in the area of online interaction. The technology allows any computer to interact directly with other computers on the network. In effect, P2P turns every computer into a client and a server, enabling a much more symmetrical and decentralized communications model for distributed applications, services and users. With the advantage of working on heterogeneous platforms across domain/service boundaries and without any centralized control/support, P2P technology promises a much better model for electronic interactions than the traditional restricted server/client one. In fact, P2P reflects society better than other types of computer architectures [1].

One of the main factors attracting research attention to P2P is the success of P2P file sharing networks, such as Napster, Gnutella and Kazaa. P2P file sharing allows users on the edge of network to directly access files from one another's hard drives. With its decentralized nature, P2P file-sharing networks provide a flexible and universal model for the exchange of information. This has been proven practically by the constantly increasing network traffic volume of P2P systems. A network traffic measurement at University of Wisconsin has shown that P2P file-sharing network traffic exceeds that of the World Wide Web [2].

Since their first appearance, P2P file-sharing network have evolved from music swapping only applications (such as Napster) to multi-functional systems that not only enable sharing variety of types of digital content but also support different classes of users (for example Kazaa and IMesh). Their use is no longer limited to file transfer between private users. Several projects are applying P2P file-sharing technology on enterprise and commercial platforms [18][20]. Despite the evolution of P2P to more complicated systems, there has been relatively little work done in access control for P2P networks. Most P2P file sharing networks give all peers the right to download all files and expect downloading activities to be controlled by human users. They do not provide any mechanisms to defend against malicious users or potential harmful sharing contents.

In this paper, we present an access control framework for P2P file-sharing networks, which provides P2P users better access control services whilst preserving the decentralized structure of the P2P platform. The framework extends a traditional access control model to meet the requirements of P2P file-sharing networks. The paper is organized as follow. Section 2 identifies the requirements of an access control model for P2P systems. Section 3 discusses the characteristics of P2P systems. Section 4 explains our access control framework in detail, including the overall architecture, authentication process, scoring scheme, and the download procedure. Section 5 discusses related works and compares our proposed access control scheme to these works. Section 6 gives our concluding remarks.

2. Access Control Requirements

We have identified 4 main requirements that an access control model for P2P file-sharing networks should support:

- *No centralized control or support*: Traditional access control models, such as ACL or RBAC [16], generally rely on central servers for authorization operations. This gives a central location at which access control policies can be stored and evaluated. A single authority, which identifies users, defines roles or groups and controls the access rights, obviously simplifies the management process. However, such centralized access control authority does not exist in a P2P network. In fact, a peer has a significant level of autonomy and is in change of storing and managing its own access control policies. An access control model for P2P networks must take this decentralization into account.

- *Peer classification*: Another characteristic of P2P networks is that of peer anonymity. Unlike client/server systems where machines and users are tightly coupled and binding to the legacy system of the enterprise they belong, peers in a P2P system are typically loosely coupled and provide very little information about their real-world identities. Their interacting partners are mostly unknown, unlike most other systems, where the users are known. A host peer may be contacted by previously unknown users, who request access to the system. However, from the host's point of view, users are not all the same. A P2P access control model must provide a mechanism for a host peer to classify users and assign each user different access rights, even if the users were previously unknown.

- *Encourage sharing files*: The incentive for users to join a P2P file-sharing network is the availability and richness of files the system provides. Implementing an access control system effectively means reducing the chance that users will get their desired files. This discourages users from participating in the system and reduces the number of files offered to share. Access control for P2P systems should attempt to minimize this problem. While it needs to give peers the ability to control access to their files it must still encourage them to share their files. Peers must be confident that participation in the system will give them better chance to access to the files they want.

- *Limit spreading of malicious and harmful digital content*: The open and unknown characteristics of P2P make it an ideal environment for malicious users to spread unsolicited and harmful content, such as pornography, viruses, or worms (e.g. VBS.Gnutella worm [10]). A P2P access control system should support mechanisms to limit such malicious spreading and punish those who are responsible for it.

3. Peer-to-Peer File Sharing Systems

P2P file-sharing allows any two peers in the system to directly access files from each other's systems (typically stored on the hard drives). To achieve this flexible direct access, any peer in the network has to support two essential interfaces:

– *Resource Discovery*: this interface allows a peer to find out what other peers offer to share as well as letting other peers know to what is available for sharing in its machine. Algorithms to effectively and efficiently find the needed information in a P2P network have been extensively studied in [12][13][14]. We will not discuss these algorithms further since the topic is not the focus of this paper.

– *File Transfer*: the interface for transferring files from one peer to another during the download transaction. Existing P2P systems often build the interface on top of the application layer of network protocols such as TCP/IP or HTTP.

In traditional P2P architectures, these interfaces access directly local file systems to allow client peers to retrieve their desired information in an unrestricted manner. We assume that the files owned by each user, and supplied to the P2P system, are stored on the user system. That is, there is no file storage sharing or duplication. We realize that this is a simplification but for our initial model, we will make this assumption. We will be extending this to a storage repository within a user domain and then outside of the user domain in our subsequent work. In this paper, our main objective is not to overcomplicate the system architecture but first develop an access control framework that makes use of newly developed trust mechanisms for P2P systems.

4. Access Control Framework

Peers in a P2P file-sharing network need the autonomy of controlling accesses to their files. Due to the lack of centralized infrastructure and global view of the whole P2P system, we propose an access control framework based on the discretionary access control (DAC) model [11]. This leaves the control of access rights to the discretion of the owner of the object (file). However, in our system, we cannot pre-assign access rights to users, as the user community in a general P2P network is unknown at the time of policy creation.

Our framework considers a host peer as a standalone system where shared files are objects that need to be protected and client peers are subjects who are considered to possess, or gain access rights. Files on a host peer are rated depending on their size and content; each file being assigned two thresholds which capture two access aspects. A client peer who wishes to download a file needs to have its two access values both equal to or greater than the corresponding thresholds of the file. The access values are relative and assessed on a peer to peer basis. They computed from combinations of four different scores: *direct trust*, *indirect trust, direct contribution* and *indirect contribution*. The client peer is responsible to collect recommendations that contain the information needed to evaluate its access values for a particular host. After each download transaction, direct trust and direct contribution of both the client peer and host peers are updated accordingly to the *satisfaction level* of the transaction, which then affect the future evaluation of the access values between these two peers.

On the other hand, our access control framework not only provides access control services for individual peers but also focuses on collaboration of controlling access between peers. A client peer's indirect trust and indirect contribution are calculated based on how access control systems on other peers evaluate their interaction with the client peer. The scores make peers inter-dependent in term of granting access to a particular sharing file in the network. Peers also collaborate to isolate malicious peers. Furthermore, rating certificates are standardized and each peer implements an access control interface that enable peers to talk one to another over the network.

4.1. Overall Architecture

In our proposed model, an extra authentication and access control layer is introduced which sits between the network interfaces (i.e. the resource discovery and file transfer) and the local file system interfaces (see Figure 1). This layer is responsible for authenticating partner peers, calculating access values, granting access to files and updating local access control policy. It may also give local users explicit rights to grant or deny any peers' requests to access to the sharing files. An additional access control interface is also provided for peers to exchange their access control information such as to get and to issue the rating certificates.

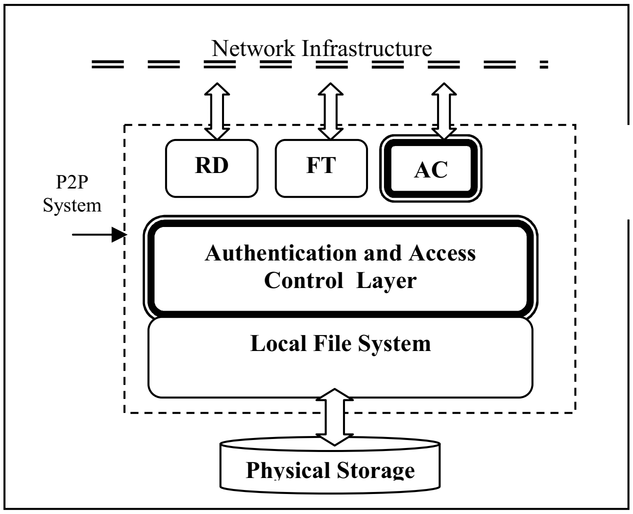

Figure 1: *Overall layer architecture of a peer with access control implementation*
(RD, FT and AC denote resource discovery, file transfer and access control respectively)

4.2. Authentication

In any security system, authenticating a subject is an essential step before authorizing the subject for any protected operation. In our framework, a peer is equipped with a 128-bit GUID (global unique identifier [17]) number and a pair of public/private keys. The peer gives out the GUID and the public key as its identity and uses the private key for authentication. We avoid using the common PGP style identifier [24] in which a subject is equipped with a plaintext name and a public key pair because there is potential to identity conflicts where two peers may choose same plaintext name and same public key.

In a P2P transaction, the authentication must be mutual. This means both a client peer and a host peer need to be authenticated with each other. The authentication process is initialized by the client peer who wishes to make contact. The authentication procedure is as follows. First, the client sends an authentication request, containing its GUID and public key together with a secret encrypted by the host's public key, to the host. This supposes that the client knows who it is going to interact, that is, it follows a successful resource discovery. Upon receiving the request, the host checks in its database to see if the client has previously contacted it. If so, there will be some existing trust information. If the client has not previously contacted the host, then the host will create a database entry for it. The host then carries out some form of authentication protocol. Our architecture supports an authentication protocol based on SSL [23].

The host peer maintains a local database to track client peers' records, not only for authentication purposes but also for access control purposes such as trust and contribution calculation. The host peer may limit the number of peers kept in its database in order to prevent the database growing to unmanageable proportions. This could be achieved, for example, by discarding the records of peers who have not contacted the host for a given period of time.

4.3. Scoring System

As mentioned in section 2, a host peer needs to classify its client peers in order to provide them different access privileges. Our model uses a scoring system to differentiate peers based on their behavior in the P2P network.

Hence, after completing the authentication process with the host, a client peer is required to supply its *rating certificates* for the host to calculate the client's relative access values. Two access values that the host assigns to a client peer are subjective. They indicate how the host peer perceives the client's trustworthiness and contribution level. The former is to ensure the peer is trusted to interact with. The latter is to promote fairness in P2P network.

From the host's perceptive, there are two sources of information to compute these two values. One source is the host's direct experiences with the client. The other's other peer's recommendations based on their interactions with the client.

Therefore, the access vales are evaluated via combinations of four types of scores: direct trust, indirect trust and direct contribution and indirect contribution. *Direct trust* represents the host's belief on the client's capacities, honesty and reliability based on the host's direct experiences. *Indirect trust* represents the host's belief on the client's capacities, honesty and reliability based on recommendations from other peers. *Direct contribution* measures the contribution of the client to the host in term of information volume downloaded and uploaded between them. *Indirect contribution* measures the contribution of the client to the network in term of information volume the client exchange with other peers.

The reason behind using direct trust and indirect trust as key factors for making access control decision is due to the unknown factor in P2P network. A host peer often has no real-world information about a client peer to determine whether it should trust that client to allow it to access to the local files. The only way to evaluate the client's trustworthiness is to go through its past behaviors as experienced by the host and other peers in the network. Naturally, the host only believes other peer's recommendations to a certain extent, as it does not place total trust on these peers. Contribution scores, on the other hand, are used in the manner of a "payment" scheme; they vary depending on the amount of information a client peer shares with the host peer as well as other peers. It tries to reduce the problem of unfair trading such as "free riding" [3] by capturing the idea that "more a peer uploads its files to the network, the more likely that peer will be able to download its desired files from the network".

4.3.1. Direct Trust

Several algorithms have been proposed to calculate the trust that a peer has over another based on their direct interaction history [6][7][8][9]. Typically an algorithm fits in a particular scenario depending on its advantages and its limitations. In this paper, we start with Beth et al 's formula, which is used to estimate a node's trust in an open network [15]. The formula is interpreted within the context of P2P scenarios as follows:

$$\mathbf{T_{ij} = 1\text{-}\alpha^n}$$

T_{ij} denotes the trust value that peer *i* has in peer *j*. *n* is the number of peer *i*'s *satisfied transactions* with peer j (satisfied transaction is defined in section 4.4.3). *α* is the learning rate – a real number in the interval [0,1]. Consequently, T_{ij} is in the range of [0,1]. It is noted that as the number of peer *i*'s positive experiences with peer *j* increases, the trust value, T_{ij}, approaches 1. *n* starts as 0 to reflect zero trust when there is no prior interaction between peer *i* and peer *j* (that is, unknown peers). Figure 2 shows how the trust values grow with different learning sample rates, *α*. The smaller the *α,* the faster the trust value grows. Hence, the value of *α* should be chosen reasonably high enough to ensure sufficiently safe estimation.

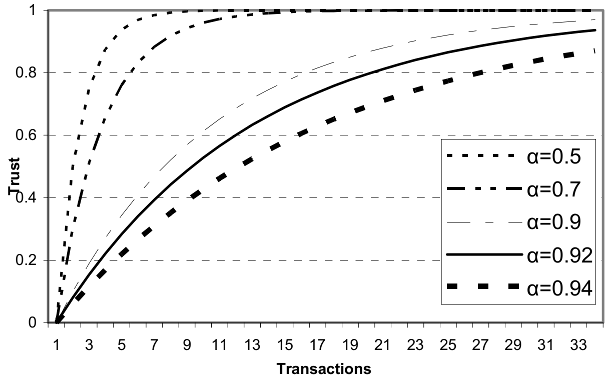

Figure 2: *Trust value against number of transaction with different learning rates*

4.3.2. Indirect Trust

In a P2P file-sharing network, a host peer often encounters a client peer that it has never met; consequently there is zero direct trust between them or the client has not interacted with the host long enough for the host to trust the client to a sufficient level. In both cases, the host has to estimate the client's trustworthiness using recommendations from other peers whom it knows. Indirect trust that the host has in the client based on recommendations is calculated as follows:

$$\mathbf{R_{ij} = (\sum_{t=1}^{k} T_{it}{*}T_{tj})/k}$$

R_{ij} denotes the indirect trust of peer *i* in peer *j*. *k* is a number fixed by the host. If the client provides recommendations from more than *k* peers, then the host uses only the *k* highest recommendations. If less than *k* recommendations are supplied, then the division is still by *k*, resulting in an understandably lower indirect trust level. The indirect trust will be a quantity in the range 0 to 1. The recommendations involve only a single level of indirection; that is, they must be from peers who have interacted with both peer *i* and peer *j*. T_{it} denotes the direct trust value of peer *i* in peer *t*, and T_{tj} of peer *t* in peer *j*. Since the direct trust values T_{it} and T_{tj} are in the interval [0,1], the indirect trust of peer *i* in peer *j* (based on peer *t*'s recommendations) is always less than T_{it} or T_{tj} (see Fig. 3)

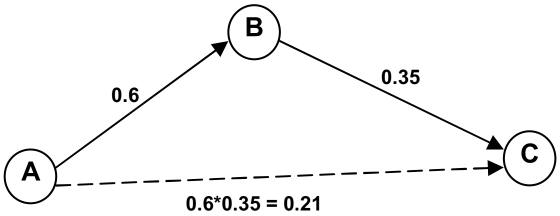

Figure 3: *Example indirect trust of peer A to peer C calculated via peer B's recommendation*

There are three main reasons why we divide by *k* to get an average indirect trust and not by the number of recommendations supplied (in the manner that other systems [6][7] do).

- In our framework, a client submits other peers' recommendations to a host in the form of rating certificates (defined in 4.4.1). Submitted recommendations are therefore at the client's selection. If we simply average the indirect trust value, the client could submit only one highest recommendation to get the highest possible indirect trust value.
- It is reasonable to expect that the greater the number of peers that the client has interacted with, the more recommendations it has and the higher its indirect trust as rated by the host peer.
- Allowing the host to set *k* allows it, to some extent, specify a required number of recommending peers. In general, this can vary from peer to peer and depending on the type of transaction.

In [15], Thomas and Malte show that an indirect trust value can be derived via a recommendation path. For instance, *A* has some trust on C_0; C_0 has some trust on C_1, …, C_{n-1} has some trust on C_n and C_n has some trust on *B*. Therefore *A* should trust *B* to some degree based on recommendation path *C*. However, a recommendation path involving two or more nodes is not favored in our framework as it is implementation expensive and complex to derive a very insignificant trust value (as the derived trusts are product of direct trusts in the range of [0,1)). Furthermore, the transitivity rule in general may not be applicable.

4.3.3. Direct Contribution

Direct contribution is measured in megabytes. It indicates the relative transferring volume of shared information from the client peer to the host peer over their interaction history. Hence direct contribution is defined as

$$\mathbf{Q_{ij} = D_{ij} - D_{ji}}$$

Where Q_{ij} is the direct contribution score of peer *j* to peer *i*; D_{ij} denotes the amount of information (in megabytes) that peer *i* has downloaded from peer *j;* similarly D_{ji} denotes the amount of information (in megabytes) that peer *j* has downloaded from peer *i*. It is noted that Q_{ij} may have a negative value if peer *j* has downloaded from peer *i* more than peer *i* has downloaded from peer *j*. Furthermore,

$$\mathbf{Q_{ij} = -Q_{ji}}$$

Direct contribution captures the notion of whether the host "owes" the client downloads based on past interactions. It does not matter that a long exchange of downloads may make this quantity to be zero – the trust from such interaction would be captured in the direct trust value.

4.3.4. Indirect Contribution

Similar to direct contribution, indirect contribution is measured in megabytes; it indicates the relative transferring volume of shared information from the client peer to other peers. The data source for computing the client's indirect contribution volume comes from the recommendations that the client has to provide the host prior requesting a file. However, from the host's perspective, recommendations from different peers must be weighted differently; depending on its trust level on the recommending peer. Hence, the formula to evaluate one's indirect contribution is as follows.

$$\mathbf{P_{ij} = \sum_{t=1}^{k} T_{it} * Q_{tj}}$$

P_{ij} denotes the indirect contribution score of peer *j* from peer *i*'s point of view. Q_{it} is the direct contribution score of peer *i* to peer *t*. *k* is number of recommending peers.

Even though the indirect contribution value tries to capture the contribution of one peer to the P2P network, it is still assessed on a peer-to-peer basis. This is partly due to the lack of a global view of a peer's behavior (as there is no centralized support or control). The above formula to evaluate indirect contribution of a client to a host reflects the view that it is realistic for the host not to be concerned about what the client has done to each peer in the network but to focus on what the client has done to the peers the host knows and trusts. In this sense, this reflects the real world.

Using the direct trust as a weighted factor in the indirect contribution formula helps to neutralize the effect of colluding parties, where a group of peers lie about their contribution against each other, thereby derive greater benefit from the network. In our case, if two peers lie about the contributions to one another and they have not contributed much to the network, then the indirect contribution values of these two peers to a third peer are relatively small (since the third peer does not have much trust in any of them).

It is noted that the contribution scores, both direct and indirect in our framework give P2P users incentive to share their resources. In this sense, it resembles a payment scheme (e.g. [5][21][22]), where users have to make some contribution in exchange for the benefit they receive from the network. In our case, a peer's contributions and benefits are defined in terms of the amount of digital content (in megabytes) it uploads to and downloads from the P2P network.

4.3.5. Granting Access

Before making a file available for sharing, a host peer defines the access control policy for the file according to the file's size and content such as its sensitivity or uniqueness. Unlike the conventional access control model, a file's access control policy in our system does not assign access rights to any specific user because users are not known in advance. Instead, it specifies two thresholds for the file: one for trust value and the other for contribution score. Trust threshold, denoted as A_{th}, is a quantity between 0 and 1. Contribution threshold, denoted as B_{th}, is a real number and measured in megabytes. Now the access rule can be formulated as follows: Any client peer who has its overall *trust* value and *overall contribution* score equal to or greater than the corresponding thresholds can access to the file.

The overall trust value, denoted as *A*, that a host has on a client peer is a weighted summation of direct trust and indirect trust. The overall contribution score, denoted as *B*, is a weighted summation of direct contribution and indirect contribution. The host sets these weightings in a variety of ways. The weightings may be set the same for all of a host's files, for sets of the host's files, or may be set on an individual file basis. Regardless of how they are set, the four weightings, C_T for direct trust, C_R for indirect trust, C_Q for direct contribution and C_P for indirect contribution must satisfy

$$\mathbf{1 = C_T + C_R}$$
$$\mathbf{1 = C_Q + C_P}$$

Hence, overall trust and overall contribution of client *j* to host *i* regarding to a file are

$$\mathbf{A_{ij} = C_T{*}T_{ij} + C_R{*}R_{ij}}$$
$$\mathbf{B_{ij} = C_Q{*}Q_{ij} + C_P{*}P_{ij}}$$

A_{ij} must be greater than or equal to A_{th} (threshold) and B_{ij} must be greater than or equal to B_{th} (threshold) for the client to be granted access. Depending on files, the host can set minimum required value for T_{ij}, R_{ij}, Q_{ij} or P_{ij} for better protection. For example, $T_{ij} >= 0.5$ means that client *j* must have interacted with host *i* and gain host *i*'s trust of at least 0.5 to be granted access to the file.

In principle, a host peer sets the trust thresholds for its files according to the files' quality (such as sensitivity and value of the content) and sets the contribution thresholds based on the files' size. The rationale is that a client peer can only download the files of similar quality to the files it shares with other peers and in our case, the file size matches its contribution. However, a host may set these thresholds differently based on its policies.

Tailoring these weightings on a file basis gives the host the needed flexibility. The host is flexible to judge a client based on its direct experiences or to listen to recommending peers. Furthermore, it is important to note that a host is also able to omit components in the formula if it chooses to. For instance, if the host wishes to discard indirect trust completely, it can discard it by assigning a zero weighting thereby only taking the direct trust into account.

4.4. Trust and Contribution Score Management

4.4.1. Rating Certificates

A rating certificate is used as a means to deliver a recommendation. It contains the direct trust value and the direct contribution score of the recommending peer on the recommended peer. The recommended peer is responsible to hold and store the certificate. The recommending peer issues and signs the certificate using its private key in order to protect against any unauthorized modification while the certificate is in the recommended peer's possession. The rating certificate also contains an expiry date. This prevents the recommended peer from recycling good rating certificates while its current rating with the recommending peer may be actually lower. When a certificate is going to expire, it's the recommended peer's responsibility to contact the recommending peer for renewing or extending the validity of the certificate. When two peers interact, the certificates that one issued to another are automatically get updated.

Generally, a rating certificate consists of following fields: recommending peer's identity, recommended peer's identity, the direct trust value, the direct contribution score, an issuing date and time, an expiry date and time and the signature. An example of a rating certificate is shown below.

```
<RatingCertificate
Signature='T2hgS678K8gbxe690'/>
        <ExpirationDate Date=02.06.2004
Time='15.59'/>
        <Recommending GUID={ADE6444B-C91F-
4E37-92A4-5BB430A33340}
         PublicKey='xxxxxxxxxxxxxxxxxxx'/>
        <Recommdended GUID={BCB80276-4807-
11d2-9717-00C04F79E98B}
         PublicKey='xxxxxxxxxxxxxxxxxxx'/>
        <Value DirectTrust = 0.65 DirectContribution
= 259/>
```

```
        <IssuseDate Date=02.05.2004
Time='15.59'/>
</AuthenicationCertificate>
```

The above design and structure of rating certificates in a P2P system bring a number of advantages, including the following:

- The client peer can be forced to be responsible for the task of collecting rating certificates since it is the one who benefits from an interaction.
- The host peer can assess the client peer independently from the recommending peers' availability.
- Helps to avoid a large increase in traffic network as the host does not have to query to other peers for the client's rating every time the client requests for access.
- Expiration date in a rating certificate can be used to reflect the varying nature of a peer's trustworthiness over time.
- Extra information can be added to the certificates in the subsequent upgrade versions of our framework without completely altering the interface of the existing system.

4.4.2. Local Storage

In order to make the system scalable, a peer in our system does not keep track of the transaction details with other peers. Instead, a peer just stores two sets of rating certificates in it local database. One set is the received certificates in which the peer itself is the recommended peer. It is the peer's self interest to retain these certificates for future downloads. The other set is the certificates which the peer issued to other peers. For each peer it has interacted with, the peer needs to keep a copy of the latest certificates it issued to that peer for validation purpose and for trust and contribution score updating.

In addition, we propose that a peer maintains a black list of peers who it believes to have committed malicious acts. The idea is that the peer will refuse extending rating certificates and doing business with any peers that are in this black list. To further isolate the malicious peers, peers can exchange the black list with its trusted partners.

Hence, given the limited amount of storage needed to keep these certificates and the black lists, the local database is relatively small even for interactions with numerous peers. Furthermore, the use of public key based infrastructure makes our architectural approach scalable.

4.4.3 Transaction rating

After completing a downloading operation, a client peer has to issue the host peer a new rating certificate, updating its direct trust and direct contribution score with the host based on the transaction's satisfaction level. We envisage that failing to do this would result in the client being added to the host's black list. We expect a peer to conform to certain standard norms of the P2P network. In our system, there are 2 satisfaction levels for a transaction: *satisfied* and *unsatisfied*.

A satisfied transaction gives the host better direct trust value. This could help the host download its desired files from other hosts, for which the host does not have enough access rating before. The formula to update direct trust was discussed in section 3.4.1.

$$\mathbf{T = 1\text{-}\alpha^{n}}$$

where n is number of satisfied transactions. However, there is no universal way to define a satisfied transaction. Different peers have different expectations on different aspects of a transaction. The overall evaluation of a transaction is a combination of evaluations of all aspects related to the transaction. For the sake of simplicity, we evaluate a P2P file-sharing transaction based on two main aspects: download speeds and file quality. These two aspects are the biggest contribution factors to the performance and success of the P2P network. A transaction is then defined as *satisfied* if the download speed is *acceptable* and the quality of the download file is rated as *fair* or better by the client.

The download speed of a transaction is acceptable if its average transferring rate is greater than a threshold rate. Otherwise, it is unacceptable. Transferring rate of a transaction can be computed instantaneously after each transaction. We define 5 levels for assessing the quality of a shared file: *good*, *fair*, *poor*, *corrupted*, and *harmful*. Human users rate the file according to its content. There can be several issues here. For instance, human evaluation of the quality of a file could be a long process compared to the instantaneous assessment of download speed evaluated automatically. Nevertheless, the host often expects an immediate response about its rating from the client after each transaction. Our solution is to evaluate aspects of a transaction separately. An interaction is first rated by its download speed so that a new direct trust value can be calculated immediately. If the download speed of a transaction is acceptable:

$$\mathbf{T = 1 - \alpha{}^{\wedge}(n+1)}$$

If it is unacceptable,

$$\mathbf{T = 1 - \alpha{}^{\wedge}(n-1)}$$

The client issues to the host a rating certificate which contains the new trust value and the updated contribution score. The expiry date in the rating certificate is set in such a way that human users could finish evaluating the file quality by that time. This will also give the opportunity to update the certificate later. In the case, where the certificate expires and the human user has not finished

rating the quality of the file, the certificate will be extended till the next expected date.
We will be discussing such issues in a separate paper which focuses on the different policies that needs to addressed and the various design options associated with them.

The file quality rating affects the direct trust value as follows:

- *Good*: **$T = 1 - \alpha^{(n+1)}$**
- *Fair*: **T unchanged**
- *Poor*: **$T = 1 - \alpha^{(n-1)}$**
- *Corrupted*: **$T = 1 - \alpha^{(n/2)}$**
- *Unknown:* **$T = 0$**
- *Harmful or malicious*: Add to the Black List

If the file is rated harmful or malicious, the client will add the host peer into its black list.

Note that with our rating system, peers sharing poor or corrupted contents will have their trust reduced in the downloading peer. This discourages peers from indirectly or indirectly distributing such files. Moreover, malicious peers will be blacklisted and isolated from the P2P network via the blacklist exchange protocol.

4.4.4. Validation

It is understandable that our framework will bring some performance overheads to the P2P networks. We believe that the main overhead is the validation work which includes checking the validity of signatures in the rating certificate that a client submits to a host prior to their interaction and checking with the certificate issuer to ensure that the certificate is the latest issuing one.

However, the host does not necessarily validate all the rating certificates that the client submits to it. Instead, it can either validate the most weighted certificates in the final scores or validate randomly some of the submitted certificate. Upon finding any invalidity, the host classifies the client as a malicious peer and adds it to the black list. Hence, any peer trying to fake the rating certificates or to recycle its old rating certificates faces a risk of being isolated from the network.

4.5. Interaction Procedure

Figure 4 represents procedures in a typical interaction between a host and a client in our framework. An interaction generally consists of three phases: preparation phase, transaction phase and feedback phase. Firstly, the preparation phase involves the authentication process, and the trust value and score calculation. The client has the choice of selecting rating certificates which give the best indirect contribution score and indirect trust value. Secondly, the transaction phase allows the client to download one or more files from the host. Each download is wrapped within a transaction, which includes the step of updating the client's and the host's interaction score. Finally, the feedback phase consists of judging the interaction based on download speed and file quality factors and issuing rating certificates as discussed in section 4.4.

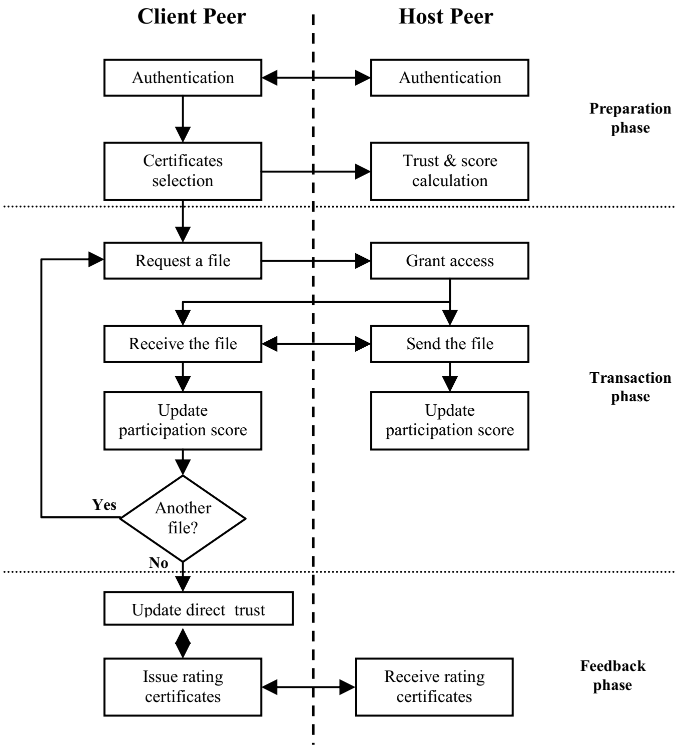

Figure 4: *Flow chart of an interaction between a host peer and a client peer*

It can be seen from the whole interaction procedure that the client plays an active role in every phase: from initiating the interaction to collecting rating certificates to updating direct trust and issue rating certificates. The host does a minimum amount of work and gets all the required information from the client and from its own database to make the decision. So our scheme adopts more of a "push" approach. We believe that this is appropriate because the design principle is that the host should not waste much of its resources (such as network bandwidth and CPU cycles), which is primarily beneficial to the client (which is obtaining the files).

5. Discussion and Related Work

Our proposed framework integrates aspects from three research areas, namely trust and reputation models, fairness based participation schemes and access control. These are applied to P2P file-sharing systems.

In each area, there have been several previous works, which address one or more problems in P2P systems. We provide a brief comparison of our scheme with some of the relevant previous works.

In [6], Yao and Julita use Bayesian networks to develop a trust and reputation model for P2P file-sharing networks. The model allows a client to evaluate file providers' trustworthiness based on its own experiences and recommendations that it queries from other peers. Hence, a transaction is made with the most trustworthy provider in the list. There are two main drawbacks with this approach. Firstly, the model does not scale well since a large database is required for each peer to keep track of all peers it has interacted with – our storage requirements are very much less. Secondly, in a real P2P file sharing network, a client peer (the beneficial side) rarely has many choices of providers with respect to the selection of downloads. Similarly, trust in [7] is built on direct experiences and reputation queries. In this model, a peer maintains a binary trust vector for every other peer it has dealt with in the past. Each 1s or 0s in the vector indicates a successful transaction or an unsuccessful transaction, respectively. Because the model is architecturally similar to Yao and Julita's one, it faces similar drawbacks. Trust protocol and trust data structures have been analysed in [8] and [9]. However, they still rely on instant recommendation queries to collect "proof-of-interaction" for decision-making. In this case, the accuracy and the response time of the trust evaluation results could be affected by the reliability of the physical network.

Regarding payment and fairness work in P2P, [4] introduces two architectures for fair sharing of storage resources in P2P systems. This paper particularly focuses on solving the problem of minority collusions – where two or more peers form a conspiracy to lie about their usage. Kazaa tries to encourage users sharing their file by giving a participation rating corresponding to the quantity of information that they uploaded Their implementation of the above approach has two major defects: storing rating score on local machine opens a way for users to cheat; users can share poor or corrupted files just for better rating.

As far as we are aware, there are three major access control models for P2P that have been proposed. Firstly, an access control system for mobile P2P collaborative environment is presented in [5]. The system provides access control services in mobile teamwork platform (MOTION) in which team-members communicate in a P2P manner. Secondly, the RBAC model is extended to support access control in a controlled P2P environment [18]. The environment contains a manager who facilitates provisioning capacities for use of resources and control usage of resource on behalf of a peer. Finally, [19] builds a multi-layered platform based on public key infrastructure, which allows peers to communicate securely. They are designed purposely for very specific scenarios and are based on a centralized approach, whereby central or trusted third parties monitor and manage access control.

As mentioned above, our framework which integrates trust, access control and fairness based participation schemes leads to a number of advantages in a P2P environment.

- A peer cannot modify its trust and contribution rating even though it stores the rating locally in form of rating certificates.
- Harmful, corrupted file providers can be punished and isolated. Bad content providers will have their accessibility to the system reduced.
- A peer has to actively share its files to gain enough access points to download its desired files.
- A client peer seeking to download has to do most of work related to the transaction.
- Access control services are provided to P2Psystems without affecting their P2P characteristics.
- Peer to peer based evaluation assures that a party of colluded peers does not obtain any significant benefit in terms of access privileges with other peers.
- The summation (as opposed to averaging) of recommendation trust helps to reduce the problems of collusion and non-reporting of trust rating certificates.

6. Concluding Remarks

The proposed trust based access control framework satisfies the four requirements of access control for P2P file-sharing systems that we identified in section 2. By extending the discretionary access control model, P2P's decentralized properties and peers' autonomy are preserved while enabling and maintaining collaboration between peers. The trust model and score systems help to classify both known and unknown visitors according to their trustworthiness and contribution. Hence, appropriate access privileges can be assigned to each visitor accordingly. The implemented contribution scores work effectively as a payment scheme; giving incentive for users to share their resources and safeguarding the fairness of service exchange in a P2P system. The proposed mechanisms for evaluating a transaction not only help to differentiate poorly performing peers from good ones but also ensure that malicious peers are punished and isolated. Although we have designed our trust based access control framework to work specifically with P2P file-sharing networks, the framework is sufficiently general so that it can apply to other P2P and other decentralized applications without major modifications.

Our future work includes refinement and implementation of the proposed trust based access scheme in a real P2P file-sharing system. Refinement of the trust scheme will take into account scenarios such as when a peer starts by offering an excellent service, and then becomes malicious by distributing malware and in punishing harmful content providers. We are currently in the process of implementing a test-bed system. We hope to use this test-bed to study the performance the characteristics such as how the system behaves with one or malicious peers and the effectiveness of our scheme in encouraging the secure and trustworthy P2P content sharing and practically measuring the amount of computational overhead involved. Following this, we are planning to extend this scheme to develop a comprehensive trust based access scheme for distributed file storage system.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments and suggestions.

References

[1] D. Clark, Face-to-Face with Peer-to-Peer networking, *IEEE computer*, Jan 2001.

[2] Evangelos Markatos, Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella, *CCGrid'2002*, May 2002.

[3] E. Adar & B. Huberman, Free riding on Gnutella, Technical report, *Xerox PARC*, Aug 2000.

[4] T.W. J. Ngan, D. S. Wallach & P. Druschel, Enforcing fair sharing of peer-to-peer resources, *IPTPS'03*, Feb 2003.

[5] *P. Fenkam, S. Dustdar, E. Kirda, G. Reif & H. Gall,* Towards an Access Control System for Mobile Peer-to-Peer Collaborative Environments, *WETICE'02*, Jun 2002.

[6] Yao Wang & Julita Vassileva, Bayesian Network Trust Model in Peer-to-Peer Networks, *AP2PC 2003*, July 2003.

[7] Selcuk, Ali Aydin and Uzun, Ersin and Pariente, Mark R.. A Reputation-Based Trust Management System for P2P Networks, *CCGRID2004,* April 2004

[8] Aameek Singh & Ling Liu, TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems, *ICP2PC2003*, Sep 2003.

[9] Karl Aberer & Zoran Despotovic, Managing Trust in a Peer-2-Peer Information System, *ACM CIKM*, Nov 2001.

[10] Symantec. http://securityresponse.symantec.com/ avcenter/- venc/data/vbs.gnutella.html.

[11] J. McLean, The Specification and Modeling of Computer Security, *IEEE Computer*, Jan 1990.

[12] B. Yang and H. Carcia-Molina. Efficient Search in Peer-to-Peer Networks, *ICDCS* 2002, Jul 2002.

[13] A. Crespo and H. Garcia-Molina. Routing indices for peer-to -peer systems. *ICDCS 2002*, Jul 2002.

[14] L Liu, KD Ryu & KW Lee, Keyword Fusion to Support Efficient Keyword-based Search in Peer-to-Peer File Sharing, *CCGRID 2004*, April 2004.

[15] Thomas Beth , Malte Borcherding & Birgit Klein, Valuation of Trust in Open Networks, *ESORICS 94*, Nov 1994.

[16] Matt Bishop, *Computer Security : art and science*, Addison-Wesley publisher, Boston, 2003.

[17] D. Box, *Essential COM*, Addison Wesley, 1998.

[18] Joon S. Park & Junseok Hwang, Role-based Access Control for Collaborative Enterprise In P2P Computing Environment, *SACMAT*, Jun 2003.

[19] W. Kim, S. Graupner, & A. Sahai, A Secure Platform for P2P Computing in the Internet, *HICSS'02*, Jan 2002.

[20] BadBlue Enterprise, http://www.badblue.com /prodee.html.

[21] K. Lai, M. Feldman, J. Chuang, & I. Stoica, Incentives for Cooperation in Peer-to-Peer Networks, *Workshop on Economics of Peer-to-Peer Systems*, June, 2003.

[22] R. Buyya, D. Abramson, J. Giddy & H. Stockinger, Economic Models for Resource Management and Scheduling in Grid Computing, *CCPE Journal*, May 2002

[23] Atkinson, R., Security Architecture for the Internet Protocol, *RFC 2401*, Nov 1998

[24] www.pgpi.org, June 2004