Extending Trust in Peer-to-Peer Networks

Stephen Clarke, Bruce Christianson, and Hannan Xiao

School of Computer Science, University of Hertfordshire, UK
{s.w.1.clarke,b.christianson,h.xiao}@herts.ac.uk

Abstract. This paper presents a way of reducing the risk involved with downloading corrupt content from unknown (and hence untrusted) principals in a P2P network. This paper gives a brief overview of the need for trust in P2P networks, introduces a new notion called trust*, and shows how this may be used in place of the conventional notion of trust. Finally, we apply trust* to the Turtle P2P client and show how the integrity of downloaded content can be guaranteed without assuming that trust is transitive.

1 Introduction

Peer-to-peer (P2P) based networks are widely used on the Internet to enable file sharing, streamed media and other services. With a traditional client-server based network, many clients connect to a fixed server. Whereas P2P clients are all considered equal and connect directly to each other. Because of this topology, tasks such as sharing files and other resources can be more efficient as a client can connect to many other clients and download content simultaneously.

Much of the content currently distributed via P2P networks is either illegal or violates copyright laws in some way. However, there are also many legitimate reasons why content might be distributed using P2P, and there is copyright-free content also available such as open source software. P2P protocols such as BitTorrent enable sharing of very large files such as operating systems, and many Linux based distributions are downloadable in this way in order to lower the load on an individual server.

P2P networks have many advantages such as scalability, and due to there being no centralised server, network loads can be easily balanced. However, for the same reasons, a problem with P2P networks is that all peers are regarded as equal and there is no real way to moderate content. Anyone can use a P2P client and share any files they wish. Malicious users can easily insert incorrect files into a network which are searchable by other clients and will therefore propagate further. Even non-malicious users might be unaware that they are serving incorrect files from their computer. To counter this, hosts might publish an MD5 check-sum on their website. However, this is unlikely and it is the user's decision whether and how they actually verify this, and getting hold of the correct checksum leads us back to the initial problem. Also, this approach assumes that the trustee is the original source and not just a middleman provider.

This paper describes how a new concept called trust* [3] can be applied to P2P networks to guarantee the integrity of files being shared. This paper uses the Turtle P2P client [11] as a basis on which to discuss the concept, although trust* can be applied to various other P2P clients. Turtle enables files to be shared among friends (people whom you know in the real world) in the hope to improve safety and overall integrity of the shared content. However, friendship isn't transitive. Trust* aims to reduce the perceived risk involved when sharing files over multiple hops with unknown principals. Trust* achieves this by providing incentives to act correctly and deterrents for acting maliciously or incompetently.

2 Extending Trust

This section briefly describes the concept of trust* [3]. The main purpose of trust* is to allow unknown principals to interact whilst at the same time lowering the perceived risk incurred by transitively trusting or relying on reputation (particularly when the intention is to use a client *once* and never again).

In the real world, this is often achieved by using an intermediary as a guarantor. An example of this is letting houses to students, where landlords require a guarantee against a particular tenant. The guarantor trusts the tenant and the landlord trusts the guarantor so the landlord has shifted the risk of not receiving the rent to the guarantor. The landlord believes that he will always get his rent whether it be from the tenant or the guarantor.

Trust* is based on the electronic equivalent of the real world guarantee solution. Say that Alice needs to trust Carol about something and doesn't personally know or trust Carol. However, Alice trusts Bob who in turn trusts Carol to do whatever it is Alice needs her to do. In order to change Alice's perception of the risks involved, Bob could guarantee to Alice that Carol will act as intended and offer Alice compensation if Carol doesn't. The concept of "extending" trust in this way by using localised guarantees is what we call a trust* relationship.

The trust*er (Alice) can then act as if they trust the trust*ee (Carol) directly. In order to shift the risk, forfeit payments are used. All forfeits are paid locally; if Carol defaults then Bob must pay Alice the agreed forfeit whether or not Carol pays Bob the forfeit she owes him (and the two forfeits may be of different amounts). Failure to provide a service – or to pay a forfeit – may result in an update to a *local* trust relationship; for example, between Bob and Alice, or between Carol and Bob. Figure 1 illustrates a typical trust* relationship.

Trust* can be composed to an arbitrary number of hops because all trust is now local and so are the forfeits. It is worth noting that trust isn't the same as trust* even in a one hop scenario; in this case, if Bob trust*s Carol to provide a service, it means that Bob trusts Carol to either provide the service or else pay the forfeit[1].

[1] Bob may believe that Carol cannot provide the service, but will always pay the forfeit.

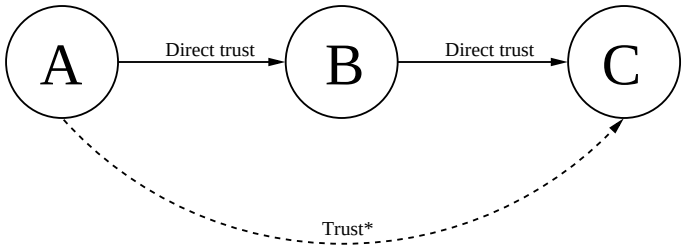

Fig. 1. A trust* relationship.

3 Trust in P2P Networks

Due to the nature of P2P networks and the likelihood that interactions are between completely unknown and untrusted principals, peers in a network need a way to mitigate the risks they would incur if they temporarily trust others. The risks involved are likely to vary depending on what is actually being shared. For example, software should be the correct version and should not be corrupted in any way, documents should be authentic and music should be licensed.

There are many security and trust issues related to P2P networks [1, 5, 9, 13] and the trustworthiness of others is normally gauged using some kind of reputation system [8, 12]. However, reputation systems have a vital flaw; they imply that trust is always transitive [6] which can be a bad assumption [2]. Assume a user wants to determine the risk involved if they were to trust another (eg. to provide a described service) by looking at their reputation rating. This might contain comments and ratings left from previous transactions. It is unlikely that the user looking knows (or trusts) the other users who have left the comments. Also, reputation systems are prone to threats such as Sybil attacks [4] where the same user can operate under many pseudonyms. But even if a user does know and trust the people who left the comments, they will still be transitively trusting the service provider in question.

According to Jøsang *et al* [7], transitivity is possible with the correct combination of the referral and functional variants of trust. However, trust* allows the risk involved to be underwritten, even when these delicate conditions for transitivity are not satisfied. With trust*, Bob is not only making a recommendation to Alice, but also offering compensation if something goes wrong. The trust scope is decided locally between Alice and Bob when the guarantee is created. It is assumed that the final guarantor in a trust* chain will have functional trust in the end-point (or trust*ee).

Most services provided over a distributed system or network have (as in the real world) an underlying contract or agreement. In most cases, this could simply be that service X will be provided for a fee P and that the service will conform to the terms and conditions of X. In P2P networks, such guidelines do not at present generally exist and clients connect to other clients to become an equal part of the network. Peers are usually free to download anything they wish from

other peers and vice versa. There may be situations where content could be charged for or where a particular service level agreement is in place, however, it is more likely that peers in a P2P network hold a "download at your own risk" policy regarding the files that they are sharing.

Trust* can be deployed to provide the missing assurance when indirectly trusting others. For example, Carol doesn't care if someone wants to download file X and doesn't care if they are unhappy with it. However, Bob has previously downloaded files from Carol, and hence trusts that her files are of a high standard. Alice trusts Bob so Bob's guarantee reduces the risk for Alice. If Bob was wrong, he will compensate Alice with the agreed forfeit. However, in this example, Carol hasn't necessarily done anything wrong and isn't obliged to reimburse Bob. Bob however is likely to lower his high perception of the quality of Carol's files and perhaps never guarantee her again. Bob's motivation to provide the guarantee is a commission payment from Alice[2]. Bob will set the level of this commission depending on his perception of the probability of Carol defaulting[3].

4 Applying Trust* to Turtle

The Turtle client requires you to list your friends whom you trust to share files with. The Turtle protocol works by only sending queries for files to these friends, who pass on the query to their friends as their own query and so on[4]. Such queries and their results are only ever swapped within these local trust relationships. The second stage is for the original requester to choose the file to be downloaded from the list of results. The file is then downloaded locally by each peer in the chain in the same manner as the search query.

This localised trust setting is perfect for also finding routes of trust* guarantees, as the query and result route used could also make up a chain of guarantees. Extending the example to a longer chain, Alice wants to download file X and sends a query to Bob whom she trusts. Bob forwards this query to Carol whom he trusts. Carol continues to forward this to her friends. Dave receives the query, he has file X and sends back a positive response to Carol which is forwarded back to Bob and then Alice. Assuming now Alice chooses Dave's file via Bob from the list of search results and requests that it comes with a guarantee from Bob, a guarantee chain could be negotiated at the same time as retrieving the file. The scope of the trust* guarantee is also negotiated between each pair which states the terms of the guarantee and what constitutes a breach.

Suppose Alice discovers that the file X is corrupt in some way. Alice can claim the forfeit from Bob. Bob may also claim from Carol. Suppose Dave does

[2] In a commercial case, where Carol provides a service for payment, Carol may pay Bob a commission for acting as an intermediary.

[3] Provided Bob's estimate of the probability of Carol defaulting is lower than Alice's *a priori* estimate, then both Alice and Bob will be happy with the guarantee.

[4] If you have read the spam-proof application in [3], please note that the direction of trust in that case goes in the opposite direction to that described here for Turtle. Trust* works perfectly in either direction.

not care if his files are correct. So rather than Carol claiming from Dave, she is likely to stop trusting him altogether, or not guarantee against him again, or charge a higher commission from Bob in future for providing the guarantee.

Eventually, say that Dave is habitually sharing corrupt content, all principals who once trusted him are likely to never guarantee his files again. In a fair P2P system where credit or reputation is gained depending on the quantity of uploaded content, and is used to download files from others, Dave will also have trouble buying guarantees from others (or they will be very expensive for him). In this example, the commission can be thought of as an insurance payment.

Alternatively, someone might guarantee only certain types of files from another peer. For example, Carol might be happy to guarantee any of Dave's music files but considers the software that he shares as risky so Carol will not guarantee these files. Trust* can enable these fine-grained decisions to be made. Even when Carol trusts Dave directly, she can still be selective over what she'll actually guarantee.

4.1 Simulation of Trust*

In order to analyse the effectiveness of applying trust* to a P2P scenario, the model was simulated with the Repast Simphony modelling toolkit [10]. A scenario where Alice wishes to download a file from Carol was simulated. There are five possible trust* routes (via the guarantors numbered 1 to 5) and each principal holds many properties including a credit rating[5]. Two global attributes t and m define the probability of Alice being truthful and Carol sharing incorrect files respectively. The trust* protocol is invoked once every "tick" of the simulation and stops when all available routes have been exhausted. The graphs in figures 2 and 3 show the resulting credit ratings for each principal and how long each simulation ran for.

In graphs (a) and (b), where Carol has a high chance of sharing corrupt files, the simulations stop after 42 ticks. By graphs (c) and (d), the probability of files being corrupt decreases, and hence, the simulation runs for longer. By graph (d) where the corruption chance is 0%, only one guarantor is ever used and the simulation would run forever. Many other graphs show fluctuations in forfeit rates and claims etc, however results presented here are limited for space reasons. The results show that long term trust* usage implies good behaviour from all involved principals. The guarantors will only tolerate misbehaviour for so long before refusing to provide further guarantees of the offending principal.

5 Discussion

5.1 Heterogeneity and Anonymity

In order to implement the trust* relationship mechanism, whether to initiate, provide, or receive a guarantee, a way of making decisions and payments is nec-

[5] This is purely to gauge the total gains and losses of a principal.

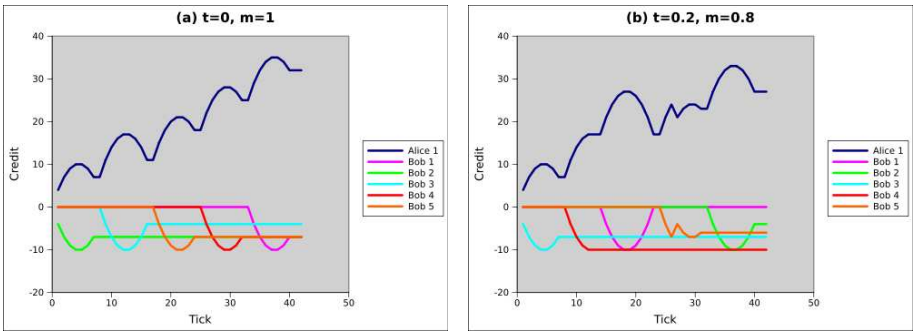

Fig. 2. Principals exhibiting bad behaviour.

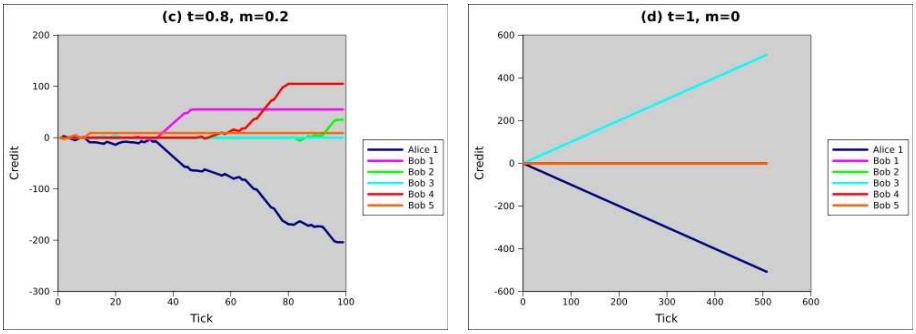

Fig. 3. Principals exhibiting good behaviour.

essary. This functionality could easily be incorporated within P2P client software. One of the advantages of our approach is that the trust management and payment systems can both be heterogeneous, due to the fact that trust (and payments) are confined or localised. If a guarantee has been made from one principal to another, any trust management and payment schemes could be used between them. At the same time, other pairs of principals might use completely different schemes.

Because of this localisation of trust, end-point anonymity can also be maintained as principals only speak to their direct neighbours. No knowledge need be gained about other principals or the schemes they might be using. Also, participants need not know whom they are downloading from.

5.2 Payment by Resource

The most obvious use of a forfeit is either to deter a principal from defaulting on what they have guaranteed or to provide a way of compensating the other

party if they do[6]. The commission payment was introduced in order to provide an incentive for a principal to act as a guarantor and can be seen as a spot price for a guarantee. A principal needing to trust* could pay this commission to a guarantor whom they trust directly. Forfeit and commission payments serve different purposes and don't need to be of the same type (or paid by the same means), although in the case of P2P networks, they could easily be.

Due to the heterogeneous nature of the localised trust between individual pairs of principals, the payments could take the form of a more immediately valuable commodity to them than a conventional micro-payment. In P2P file sharing applications, this could be the content itself. For example, credit to download further files or to buy licenses or guarantees.

6 Conclusion

This paper has presented the concept of trust* as a mechanism for guaranteeing the integrity of content or services provided over a P2P network. Trust* builds on the idea of sharing with friends in the Turtle P2P client but also guarantees the integrity of downloaded content from unknown peers derived through transitivity.

Using trust* in this way also reduces the risk involved for the downloader as they will be compensated in the worst case scenario. It therefore lowers the risk of transitively trusting others, and privacy is still maintained. This is because the guarantees and payments are confined within the same localised trust relationships as the ones that are used to communicate the actual search queries and their corresponding results. This approach therefore allows complete localisation of trust management, and the risk of trusting by referral is underwritten by the guarantees. We regard local trust management as a significantly easier problem than global reputation management, particularly in a P2P system where the majority of participants wish to be anonymous (except to their friends). As mentioned earlier, the use of trust* does not constrain the way in which local trust is managed.

Simulation of the trust* protocol shows that misbehaving principals quickly become isolated before major damage can be made. This means that threats such as a Sybil attack can be identified and the perpetrator will eventually be removed from local trust relationships. This will make it harder for them to share files in a P2P community that employs the trust* model as eventually all routes will be removed (or become too expensive).

The Turtle client was developed with an emphasis on privacy and safety of sharing files that might be of a controversial or provocative nature. Due to the localised direct trust in a trust* chain, such privacy can be easily maintained[7]. We have argued that applying trust* to P2P file sharing will also be beneficial

[6] Note that these are slightly different requirements; a lower forfeit will often suffice for the first.

[7] However, privacy is not so much of an issue when sharing open content, and in other applications where the integrity of the content is more important.

in guaranteeing the integrity of free content such as open source software or copyright-free movies.

References

1. Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 310–317, 2001.
2. Bruce Christianson and William S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the International Workshop on Security Protocols*, pages 171–176. Springer-Verlag, 1997.
3. Stephen Clarke, Bruce Christianson, and Hannan Xiao. Trust*: Using Local Guarantees to Extend the Reach of Trust. In *Proceedings of the Seventeenth International Workshop on Security Protocols*, April 2009. To appear.
4. John R. Douceur. The Sybil Attack. In *Peer-To-Peer Systems: First International Workshop*, page 251. Springer-Verlag, 2002.
5. Junjie Jiang, Haihuan Bai, and Weinong Wang. Trust and Cooperation in Peer-to-Peer Systems. In *GCC 2003*, pages 371–378. Springer-Verlag, 2004.
6. Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In *Proceedings of the Workshop of Formal Aspects of Security and Trust*, pages 9–22, 2003.
7. Audun Jøsang, Ross Hayward, and Simon Simon Pope. Trust Network Analysis with Subjective Logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*. Australian Computer Society, Inc., 2006.
8. Eleni Koutrouli and Aphrodite Tsalgatidou. Reputation-Based Trust Systems for P2P Applications: Design Issues and Comparison Framework. In *Trust and Privacy in Digital Business*. Springer-Verlag, 2006.
9. Anirban Mondal and Masaru Kitsuregawa. Privacy, Security and Trust in P2P environments: A Perspective. In *DEXA '06: Proceedings of the 17th International Conference on Database and Expert Systems Applications*, pages 682–686, 2006.
10. Michael J. North, T. R. Howe, Nick T. Collier, and J. R. Vos. The Repast Simphony Development Environment. In *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, 2005.
11. Bogdan C. Popescu, Bruno Crispo, and Andrew S. Tanenbaum. Safe and Private Data Sharing with Turtle: Friends Team-up and Beat the System. In *In Proc. of the 12th Cambridge International Workshop on Security Protocols*, 2004.
12. Ali A Selçuk, Ersin Uzun, and Mark R. Pariente. A Reputation-Based Trust Management System for P2P Networks. In *CCGRID*, pages 251–258. IEEE Computer Society, 2004.
13. Dan S. Wallach. A Survey of Peer-to-Peer Security Issues. In *In International Symposium on Software Security*, pages 42–57, 2002.