A Protocol For Storage Limitations and Upgrades in Decentralised Networks

Greig Paul
University of Strathclyde
Department of Electronic & Electrical Engineering
Glasgow, United Kingdom
greig.paul@strath.ac.uk

James Irvine
University of Strathclyde
Department of Electronic & Electrical Engineering
Glasgow, United Kingdom
j.m.irvine@strath.ac.uk

ABSTRACT

Within the MaidSafe distributed network, users are able to store mutable and immutable data, which will remain accessible on the network in a distributed hash table, essentially forever. This raises an obvious question as to how to restrict malicious parties from making use of excessive network resources, in order to ensure the storage capacity of the network is not quickly depleted by a small number of users making excessive use of storage. Such activity risks preventing legitimate use of the network. This paper explores two possible means of secure storage quota management on such a decentralised network, and presents the concept of manager verification. It then presents a protocol-based solution to allow users to purchase extra storage from other users in a decentralised network, with irrepudiable and non-forgeable agreements, which are verifiable by any third party.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms

Algorithms, Design, Security

Keywords

Decentralisation, storage quotas, digital contracts

1. INTRODUCTION

Since the very early days of peer-to-peer decentralised systems, resource management has been an important consideration, to prevent a small number of users from taking advantage of the generosity of others [7]. This is illustrated by research conducted by Sen and Wang in [11] on the FastTrack network, which showed that only 1% of network users were providing 73% of the network bandwidth needed to share files. In an earlier study carried out on the Gnutella peer-to-peer network, Adar et al. presented in [1] that 70% of users were not sharing any files, and that 1% of all users were actually providing responses to over half of network requests.

Early peer-to-peer networks were more focused on conserving bandwidth, since they were typically designed for the purpose of sharing popular files between users quickly, without relying on a central server. The MaidSafe decentralised network, while focused more on the provision of storage, faces similar challenges, where malicious users could potentially use up all of the storage capacity of the network. The most obvious means of mitigating this would be through the use of storage quotas. These however are not easy to enforce on a decentralised network in absence of a trusted authority to adminster quotas.

SIN '14 Sep 09 - 11 2014, Glasgow, Scotland UK
ACM 978-1-4503-3033-6/14/09.
http://dx.doi.org/10.1145/2659651.2659724.

2. OVERVIEW OF MAIDSAFE

Within the MaidSafe network [5], storage and bandwidth (to access the stored data) is made available to the network on behalf of a user. The network is self-managed, without any central authority to make decisions as to permitted operations, and each node is responsible for authorising the actions of other nodes. Therefore, the ability for every node to be certain of which operations other nodes can carry out is essential.

Each node (client or vault) is managed by its 4 nearest online neighbours on the network, with this proximity measured as an XOR-based distance between their addresses. Since node addresses are the SHA512 cryptographic hash of their public key, it should be computationally difficult to position a node at a desired location on the network. As nodes enter and leave the network, other nodes may become closer, and these would take over the management role. The managers of any given node should therefore be independent from the node in question.

Users store their data (in encrypted form) on vaults, which are regular computers owned and operated by other users on the network. Clearly, each user has an inherent interest in ensuring their data is correctly stored on the network, for future retrieval. Otherwise, the network itself would prove useless for its purpose of storing and retrieving data. In considering this network conceptually, it is simplest to assume that users are entirely selfish, caring only about their own data, and to require them to contribute resources for other users in order to obtain the service they desire.

A more in-depth explanation of the MaidSafe network and

its vault network security is available in [10].

3. BALANCING OF NEEDS

According to IDC analysis, only 25% of data stored is unique [3]. Likewise, the president of industrial analysts DCIG, is quoted in [4] as saying that in a corporate environment, deduplication can allow a company to store 20x more data with the same storage faclities. In an analysis of backup storage, Meyer and Bolosky [9] found that across four weekly backups, file-level deduplication achieved a saving of 72% in storage requirements. While obviously not all users will make use of MaidSafe solely to store full backup images of their computers, the ability to deduplicate data globally at file-level through MaidSafe appears likely to be able to offer significant reductions in the storage capacity necessary on the network to store files. Given MaidSafe typically holds 4 copies of each data chunk, data deduplication would appear to balance with the need for replication to ensure availability, if only 25% of the data was unique.

4. LIMITING STORAGE USAGE

Conventional, centralised, storage services often use a quota-based system of resource control, where users are permitted to store up to a certain capacity of data. [8] Beyond this point, the user would either not be able to store further data, or may be required to pay the service provider for an increased storage quota. Within a decentralised network, this is not necessarily possible without trusting a single entity to accurately and fairly enforce these quotas, and identifying which users should receive payment for storage is non-trivial.

It is therefore clear that it is necessary to have a decentralised means of restricting storage utilisation. Given the MaidSafe network has no central entity controlling account creation, we believe it is not possible to offer users a free storage quota, since such quotas could be defeated through creation of multiple accounts. We therefore propose a means of restricting storage available to users, without requiring a centralised trusted authority, and while allowing for the variation in storage quota based on contribution, be that financial or by provision of storage itself.

5. ENFORCING STORAGE LIMITATIONS

The most obvious means of enforcing storage limitations on users, in a decentralised manner, is to allow users to store a quantity of data proportional to the quantity of data they themselves are storing for other users. This ratio could be one-for-one, allowing users to store the same amount of data as they hold, or it could be non-unity, allowing users to store more or less data than they themselves store for other users (based on real-world understanding of the efficacy of global deduplication on user data). While MaidSafe have proposed the concept of offering free storage to users, without them being required to contribute resources, up to the average storage utilisation per user, we believe this too easily abused, given there is no way to restrict how many accounts one user can create - it is conceivable that a modified fork of the open source MaidSafe client could even unify these accounts, allowing users to effectively gain unlimited storage without contributing resources to the network. Naturally, this would be of significant detriment to the network. While means of restricting account creation are possible, these would ultimately end up either making it difficult or time-consuming to join and use the network (which is detrimental to the network by discouraging users), or too easy for malicious users to create more accounts.

We therefore suggest that the best means of mitigation is to only permit users to store data when they themselves are proven to be storing data. This removes the motivation for even the most selfish of users to create multiple accounts, as they would receive no extra storage by having many accounts. Within the context of the MaidSafe decentralised network, this could be carried out in 2 main ways, explained in the following sections.

5.1 Proof of Storage

Building upon work such as that by Juels and Kaliski [6] and that of Zheng and Xu [13], a proof of data storage could be used, in order to verify that a vault holds the data it claims to, and has not corrupted it. The simplest proof would be for the vault to provide the hash of data. This is unsuitable as it is easily replayed. In a DHT, data is indexed by its hash, meaning that the request to verify data must identify the data itself by its hash. As shown by Juels and Zheng though, it is possible to carry out challenge-response style proofs, which would be safe against such attacks.

A limitation of these proofs is that they require the verifier to themselves retrieve the data, in order to verify the proofs. While a verifier could pre-compute a number of challenges and responses, and then request them at a later point, this is a hindrance to ad-hoc verification. Being able to verify that a user was positively contributing to the network by storing data in their vaults, without needing to consider the data they held, would be effective and place no additional network overhead on the verification process.

5.2 Manager Confirmation

Whenever a chunk is stored on the MaidSafe network, a group of 4 nodes (surrounding the chunk's address in the DHT) act as chunk information managers, and delegate the task of holding this data to other nodes on the network. These nodes are themselves supervised by their 4 neighbours, acting as chunk managers. A user could prove their vaults' contributions to the network by telling the verifier the address of some chunks which it holds - the verifier could then enquire with the chunk managers, to ensure this chunk was reliably held by the node. By randomly checking a proportion of the chunks claimed to be held by the node, the verifier can probabilistically determine if the user is being honest in their claims of storing data based upon verifying a random selection of the claims. Since these managers monitor the online status of the vault, vault reliability can also be confirmed by the managers.

One potential risk of this approach is that it may be possible for a malicious user to cease to provide any useful storage, and take advantage of their high reputation to store a large quantity of data on the network, and no longer provide any storage. In this case, their reputation would decrease as they stored further data, as they were no longer providing storage to the network.

This verification process is illustrated in Figure 1. We believe this process to be robust, since the managers of a vault will be separate from the managers of the client. Client managers can enquire with vault managers as to the status of a vault. It does however rely on the client managers being

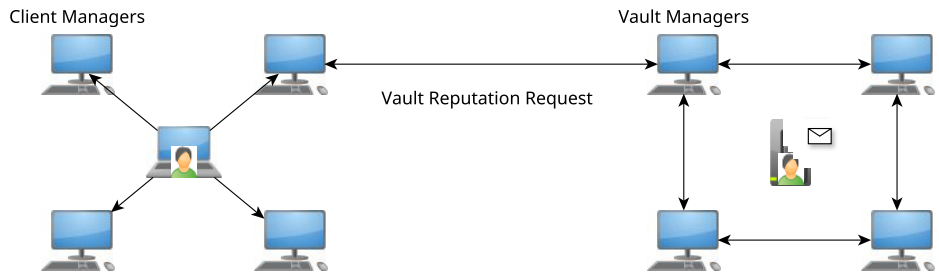

Figure 1: Storage Verification via Vault Manager

uncompromised, or a verifiable chain of consensus being in existence, which would indicate agreement with the decision of the managers.

6. VERIFICATION CONSIDERATIONS

Implementing manager-based verification would require little extra data to be retained by nodes on the network. The main security requirements of this are;

- The client requesting storage can prove it owns and controls the vault it claims is providing its storage
- The associated vault managers can vouch for it, using a non-replayable reputation query process
- The associated vault has offered sufficient storage for a sufficient period of time to offset the request
- The user is unable to "double-spend" by presenting the same set of stored data on their vault

These considerations would ensure that clients are only able to store as much data as they themselves are storing. As more data is stored, their ability to store further data would be reduced by their managers.

7. RELATION TO PREVIOUS WORKS

In previous works, storage quotas have typically been centralised in operation. For example, in the work of Druschel and Rowstron [2], storage quotas depended upon centrally issued and trusted smartcards, which contained a user's storage quota. While users could offer more storage (in exchange for an increased storage quota), and audits over provided and used storage were carried out, the network ultimately depended upon a smartcard, as "enforcing quotas in the absence of a trusted third party would likely require complex agreement protocols" [2]. Through the decentralised and peer-managed approach of the MaidSafe network, it is possible to achieve this without explicitly trusting any individual authority, since all statements are verifiable by any other user of the network.

The work of Ngan, Wallach and Druschel [12] presented the concept of auditable and decentralised quota management, however required an auditing process which required a public ledger to be permanently available, detailing a list of the identifiers of every file being held by a vault, and similarly, a list of the identifier of every file held by a client. We believe that while this is a significant improvement upon the original requirement for a smartcard in the original proposal by Druschel & Rowstron, the presence of such public audit logs poses a risk to the privacy of users on the network — users storing specific data could be targeted, and vaults providing their storage could also be identified and targeted.

We therefore propose that the ability to use a chained consensus of groups of vault managers offers the ability to carry out storage quota verification, without the need for every user to disclose the identifier for every file they hold or own. The chain of consensus ensures the probability of a vault being able to control its own managers is minimised to acceptably low probabilities (based upon SHA512 hash pre-image collisions). This method would prevent users from needing to store publicly, in a ledger, details of all data they upload themselves, or hold for others.

8. OBTAINING FURTHER STORAGE

Fundamental to ensuring the flexibility of a storage-based network is the ability for a user to store files, even when they perhaps are unable to contribute towards the storage of files for other users. In the case of centralised storage systems, users typically pay the service operator for the allocation of a larger storage quota. In the case of a decentralised network however, this is not necessarily feasible, since data is split into chunks. These chunks are distributed to many vaults, with chunk redistributed throughout the network as nodes join and leave.

As discussed by Ngan et al. in [12], it is desirable to allow users to exchange storage quotas, such that a user with a surplus of storage can agree to sell that to a user who wishes to store more data than they can themselves store. In a decentralised storage network, this matching of users means that sufficient storage will be present to ensure the longevity of the network, while offering users flexibility.

In a decentralised and trust-less network, it is necessary to identify a means through which a contract for storage could be enforced, such that the provider (rather than the buyer) would lose reputation for failing to provide the storage as agreed, and to ensure that neither party can lie about the terms of the agreed storage deal. We therefore propose a protocol, implemented on top of the existing MaidSafe network, which would permit irrepudiable and non-forgeable agreements to offset storage to be negotiated, and enforced, by the decentralised network.

8.1 Decentralised Storage Purchase Protocol

The protocol operates as follows:

- Buyer locates a provider offering storage for sale (this could be provided as a decentralised service)
- Buyer and provider negotiate a price for an agreed quantity of storage, duration, price, and offer validity
- Provider signs this offer of sale with their private key, and states a (for example) Bitcoin address in the offer
- Buyer reviews the offer, ensuring the offer is as agreed
- To complete the deal, the buyer pays the agreed funds to the provider via the agreed Bitcoin address
- The sale is verifiable by any third party able to view the contract, since the Bitcoin transaction is visible on the blockchain

The sale agreement can be made available by the buyer, allowing other nodes to verify that it has completed a transaction with the provider of the storage. The provider would be responsible for this provision, and their own reputation would be affected in the event of a shortfall affecting this user. While we use Bitcoin for payment processing, it could easily be adapted to any cryptographic currency permitting third-party verification of payments via a public ledger (like the Bitcoin blockchain), and some method of timestamping (like the block ID).

The provider is protected from a malicious buyer, since the buyer cannot forge the seller's signature. This protects the details of the offer. It also states the identity of the buyer (their public key), meaning that another user cannot attempt to claim their entitlement to storage from another user's agreement. The offer contains a validity period (to prevent a buyer from requesting an offer, then waiting until the value of storage or currency has changed, then accepting it), based upon the current block ID of the Bitcoin blockchain at the time of purchase.

The buyer is protected from a malicious provider, as the provider cannot deny the agreement exists (as the agreement was signed by the seller), and anyone can verify the agreed fee was paid, before the offer expired, thus validating the contract.

As such, this protocol allows two parties, neither of whom need trust each other, to negotiate a deal for one to provide storage on behalf of the other party. Both parties are protected from the other failing to honour their part of the deal. In the event of the provider reneging on the deal, they can be held accountable by the decentralised network, as the agreement is provable.

It is important to note that this network remains decentralised — purchasing storage from a user does not alter or influence where that data is stored — it is simply an offsetting process to ensure further storage exists in the network, to avoid storage space being depleted. Buyers are protected from price-fixing as they can purchase storage from any party on the network. As any user can already provide any quantity of storage to the network, this protocol does not reduce decentralisation, rather it increases flexibility for users.

9. CONCLUSIONS

Within decentralised storage networks, a major challenge is to ensure that resources are not abused by selfish users. With historical evidence that many users in peer-to-peer based systems avoid contributing storage or bandwidth for the sharing of files, a challenge exists for networks which rely on users to contribute storage. Without requiring a trusted third party, we have demonstrated it is possible in a peer-managed decetralised network to restrict users to only store data in proportion to what they themselves are holding for other parties. This can be tested without any significant verification overhead, by enquiring with the independent group of managers of a data-holding vault as to the quantity of storage verified as being offered by that node. We also proposed a protocol for secure, verifiable contracts which would permit users to request storage be provided on their behalf by other users on the decentralised network. This is possible without the need for trust between the buyer or seller, while allowing any third party to verify the agreement was completed, and payment was made.

10. ACKNOWLEDGMENTS

This work was funded by EPSRC Doctoral Training Grant EP/K503174/1, and MaidSafe.net.

11. REFERENCES

[1] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[2] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 75–80. IEEE, 2001.

[3] J. Gantz and D. Reinsel. The digital universe decade - are you ready? Technical report, EMC Corporation, 2010.

[4] D. Geer. Reducing the storage burden via data deduplication. *Computer*, 41(12):15–17, 2008.

[5] D. Irvine, J. Irvine, and S. K. Goo. Sigmoid (x): Secure distributed network storage. *WWRF*, 2011.

[6] A. Juels and B. K. Jr. Pors: Proofs of retrievability for large files. *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[7] F. Liu, Y. Sun, B. Li, and B. Li. Quota: Rationing server resources in peer-assisted online hosting systems. In *Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on*, pages 103–112. IEEE, 2009.

[8] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry. A fast file system for unix. *ACM Transactions on Computer Systems (TOCS)*, 2(3):181–197, 1984.

[9] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. *ACM Transactions on Storage (TOS)*, 7(4):14, 2012.

[10] G. Paul, F. Hutchison, and J. Irvine. Security of the MaidSafe Vault Network. WWRF Meeting 32, May 2014.

[11] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transactions on Networking (ToN)*, 12(2):219–232, 2004.

[12] D. S. Wallach, P. Druschel, et al. Enforcing fair sharing of peer-to-peer resources. In *Peer-to-Peer Systems II*, pages 149–159. Springer, 2003.

[13] Q. Zheng and S. Xu. Secure and efficient proof of storage with deduplication. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*, page 1, 2012.