PSH: A Private and Shared History-based Incentive Mechanism

Thomas Bocek[1], Wang Kun[2], Fabio Victora Hecht[1], David Hausheer[1], and Burkhard Stiller[1,3]

[1] Department of Informatics IFI, University of Zurich, Switzerland
[2] Research Institute of Telecommunication Transmission RITT, China
Telecommunication Technology Labs, China
[3] Computer Engineering and Networks Laboratory TIK, ETH Zurich, Switzerland
[bocek|hecht|hausheer|stiller]@ifi.uzh.ch, wangkun@chinattl.com

Abstract. Fully decentralized peer-to-peer (P2P) systems do not have a central control mechanism. Thus, different forms of control mechanisms are required to deal with selfish peers. One type of selfish behavior is the consumption of resources without providing sufficient resources. Therefore, incentive schemes encourage peers to share resources while punishing selfish peers. A well-known example of an incentive scheme is Tit-for-Tat (TFT), as used in BitTorrent. With this scheme, a peer can only consume as much resources as it provides. TFT is resilient to collusion due to relying on private histories only. However, TFT can only be applied to peers with direct reciprocity.
This paper presents a private and shared history (PSH) based incentive mechanism, which supports transitive relations (indirect reciprocity). Furthermore, it is resilient to collusion and it combines private and shared histories in an efficient manner. The PSH approach uses a shared history for identifying transitive relations. Those relations are verified using private histories. Simulations show that the PSH mechanism has a higher transaction success ratio than TFT.

1 Introduction

Peer-to-peer (P2P) systems have numerous advantages over centralized systems. Load balancing, robustness, scalability, and fault tolerance are properties that a P2P system can offer. However, challenges in P2P systems include free-riders [1], malicious peers, Sybil attacks [6], self-interest [17], and other forms of attacks [15]. Incentive mechanisms are used to address those challenges and encourage peers to act cooperatively.

A simple incentive scheme is Tit-for-tat (TFT) as used in BitTorrent [4]. In this scheme, a peer can only consume as much resources as it provides. The TFT mechanism keeps, for each peer, a history of past resource exchanges or transactions. This history is private, that is, it contains only first-hand information; consequently, peers have a limited view on transactions to peers with direct reciprocity. Thus, private history is suitable for symmetric resource interest [7], for example in a file-sharing system with many popular files. With a

TFT mechanism based on shared history, transaction information is shared and accessible by other peers, with the result that indirect reciprocity is detectable. Thus, TFT based on shared history (transitive TFT) is suitable for asymmetric resource interest. However, shared history approaches are prone to false reports and collusion [7].

This paper proposes a new incentive mechanism which is a combination of private *and* shared history. While shared history is used to propagate resource exchange information, private history is applied to verify its correctness. This approach is termed private shared history incentive mechanism (PSH). Additionally, PSH uses an efficient mechanism to propagate history by including it in resource request and response messages.

The evaluation of PSH requires two steps. First, to show that the mechanism works as expected in general. Second, to test this mechanism in a distributed environment such as PlanetLab. This paper presents the first step. PSH incentive mechanism simulations show that this approach has a transaction success ratio of up to 73% higher than TFT. Message count is, in average, 46% higher. Message size is 288% larger, in average.

There are several possible applications of PSH. In file-sharing systems with many unpopular files, there are few direct relations between peers for which TFT does not work well. In this case, PSH shows its strength. Additionally, PSH can be applied for trading resources such as computing power, memory capacity, or bandwidth in a computational Grid. In such a Grid, it is rarely the case that Grid nodes have symmetric resource interests, since those resources shared in a Grid are typically non-replicable and exclusive, i.e. resource usage diminishes the total amount of available resources [9]. PSH enables a Grid node to contribute resources to the Grid in idle times, and to consume additional resources from other Grid nodes during peak times.

The remainder of this paper is structured as follows. Section 2 discusses related work while Section 3 introduces the design of the incentive scheme. Section 4 provides the implementation details and presents results, while Section 5 draws conclusions.

2 Related Work

Incentive schemes can be divided into two groups: (1) trust-based and (2) trade-based incentive schemes [16]. With a trust-based incentive scheme, peers are encouraged to act in a certain way to gain as much trust as possible. With a trade-based incentive scheme, resources are traded and peers are encouraged to provide as much resources as they consume. This can be based on direct reciprocity as in TFT, or indirect reciprocity as in transitive TFT.

2.1 Trust-based Incentive Schemes

Kamvar et al. [11] propose a global unique trust value based on a peer's history. The EigenTrust algorithm can effectively identify malicious peers and isolate

them. The authors have showed in simulations that their approach can reduce the number of peers providing inauthentic files.

Lian et al. [12] propose multi-level TFT to achieve robust incentives. This approach is a balance between EigenTrust and TFT. The authors have implemented and tested their approach in the Maze [22] system. Multi-level TFT introduces limited indirect trust levels, to reduce the trust metric size. As in EigenTrust, multi-level TFT aggregates transitive trust values. The evaluation shows that a two-level matrix performs better than relying on private history only, while efficiently dealing with malicious peers.

Another enhancement of EigenTrust is described in [5]. The authors compared EigenTrust with the following different extensions: inverse EigenTrust, truncated PageRank, bit propagation, and badness based on BadRank. Simulations showed that EigenTrust with badness based on BadRank performs always better than EigenTrust alone with respect to authentic downloads.

A similar approach to EigenTrust is described in [14]. The authors propose a voting reputation system, in which opinions of other peers are considered. Those peers can vote on a peer's reputation. A highly reputable peer vote has a higher value. However, this approach is prone to whitewashing, and a high cost for initial joining is a side effect.

PeerTrust [20] compares the trustworthiness of peers using a new trust metric. This metric uses three important trust parameters: feedback, number of transactions, and the credibility of the feedback. The authors showed in simulations the feasibility of their approach.

2.2 Trade-based Incentive Schemes

Trade-based incentive schemes, such as KARMA [19] and the mechanism used by PPay [21] introduce a broker role. PeerMint [10] uses multiple remote peers to store and aggregate accounting information in a trustworthy and scalable way. It applies a structured P2P overlay network to map accounts onto a redundant set of peers and organizes them in an efficient and scalable manner. The scheme uses session mediation peers to maintain and aggregate session information about transactions between peers. This minimizes the possibility of collusion. However, malicious peers acting as a brokers or mediators are an open issue.

Feldman et al. [7] proposed to use the MaxFlow algorithm for collusion free data propagation. The authors suggest to modify the MaxFlow algorithm to evaluate paths in constant time. However, not all paths will be found (indirect reciprocity).

Ngan et al. [18] present an architecture to enforce fair sharing of storage resources, which is robust against collusion. The architecture uses auditing and usage records, which are publicly available. A peer can be randomly audited by any other peer. The authors show in simulations that the overhead of auditing is small and scales in large networks, and that peers have an incentive to provide correct data.

2.3 Comparison of Incentive Schemes

Table 1 shows a comparison of related work. Since PeerMint, KARMA and PPay require a broker, and PeerTrust focus on metrics, they are not included in this table. Unlike multi-level TFT, PSH does not aggregate the propagated values. PSH verifies each information directly on a peer, thus detecting collusion. The path finding algorithm presented by Feldman et al. requires many contacts with other peers, while PSH requires few contacts because transaction data is locally available, as this data is propagated with each request. The work presented by Ngan et al. uses auditing to verify resource information, while PSH tries to find a transitive resource exchange path.

Table 1. Related work comparison

Algorithm	*Aggregation*	*Transitive exchange*	*Collusion resistance*	*Local data*
PSH	no	yes	yes	yes
Eigentrust	yes	yes	no	yes
ML-TFT	yes	yes	no	yes
Feldman et al.	no	yes	yes	no
Ngan et al.	no	no	yes	yes

3 Design

PSH uses shared history to find peers with indirect reciprocity and private history to verify the reciprocity for those peers.

3.1 Requirements

Although PSH uses shared history, the mechanism shall be collusion resistant. In addition, data propagation must be scalable, robust, and fault tolerant. To allow an initial transaction, a peer may consume resources until a credit limit is reached. The credit limit must be low enough to discourage peers from creating new identities (white-washing). A last requirement is the workload to be placed on the requesting peer, preventing DoS attacks. If load is placed mostly on the answering peer, too many requests may cause an overload.

3.2 Assumptions

It is assumed that peers have asymmetric resource interests, i.e. a peer that provides resources to another peer has no interest in the resources that peer has to offer. Furthermore, each peer's public key is assumed to be known or able to be requested. Since signatures are verified directly on the respective peer, no trusted third party is required.

3.3 Algorithm

PSH behaves like TFT if two peers are about to exchange resources and previous transaction information is present in the other's private history. If it is not present, PSH looks for a path, that is, a linked list of peers with transitive history information, from the source peer to the target peer. Each of those peers in the path is requested to issue a check, that means, to transfer their signed credit balance between the source and the next in the path or the target.

Figure 1 shows the general architecture of the request / response handling in PSH. Arrows between grey boxes indicate a TCP or UDP connection. The figure shows that the workload is mainly on the requesting peer, because path searching is done on the requesting peer.

In Figure 1, peer s sends a resource request to peer t. If the request succeeds, then both private histories from peer s and t are updated as in TFT. If it fails, peer s searches for a path. If a path cannot be found, the request fails. If the path cannot be traversed, the request fails as well. If the path can be traversed and a valid check can be returned, peer s asks peer t again with this valid check.

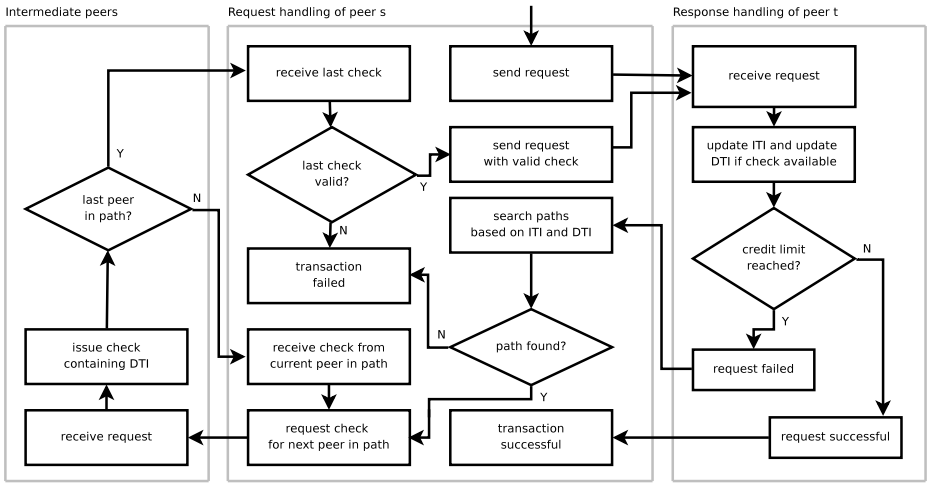

Fig. 1. General PSH incentive mechanism

3.4 Shared History Data Propagation

Each peer stores two tables of history information, a direct transaction information (DTI) table and an indirect transaction information (ITI) table. A DTI table contains information based on direct reciprocity (private history). An ITI table is based on indirect reciprocity (shared history). A DTI entry for peer x and peer y $DTI_x(y)$ is defined as the amount of exchanged resources. For a successful resource transaction from peer x to peer y, the former stores $DTI_x(y)$,

while peer y stores $DTI_y(x)$, where $DTI_x(y) = -DTI_y(x)$. Along with each request and response message, a subset of DTI entries with the highest timestamp is exchanged to avoid creating new connections. The ITI table contains accumulated DTI from other peers, *e.g.*, peer x has $ITI_y(z) = 3$, which means peer x knows about transactions between peer y and peer z. The ITI and DTI tables are used to find a path from a given source to a target. If such a path exists, then indirect reciprocity can be inferred.

Since many paths may exist, the size of a complete ITI table has polynomial complexity $O(n^2)$, where n is the total number of peers in the system. A reduction of complexity can be achieved by evaluating a limited path length L instead of $|n|$, where $L < n$. Further complexity reduction can be achieved by expiring transactions in the ITI table using a time decay function $f_{decay}(transaction)$. Therefore, not all existing paths are found.

A transaction between two peers x and y is defined as $T_x(y) = z$, z being the transaction value. Each transaction contains a timestamp. Figure 2 shows example values of DTI and ITI in a state in which transactions T have already happened. In this example, peer s is the source and peer t is the target. For better readability, the ITI table is only displayed for peer s.

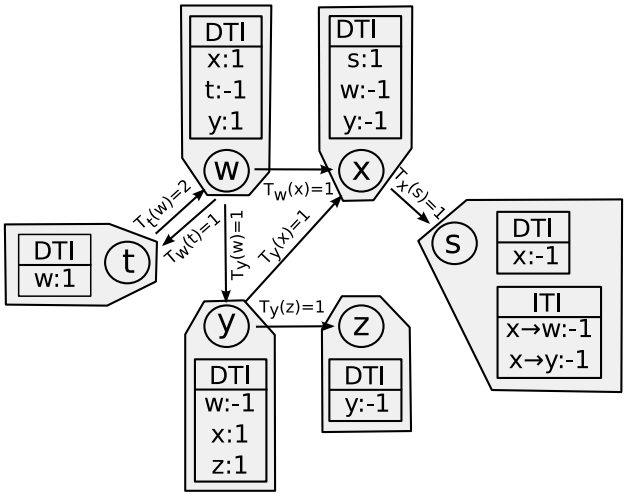

Fig. 2. Initial states of peer t, w, x, y, z, and s, showing DTI and ITI tables

3.5 Private History Verification

Once a path has been found using the shared history, the verification process starts. This process queries every intermediate peer on that path $P(s, x, \ldots, t)$, where s is the source, t is the target and $x, \ldots$ are intermediate peers, to issue a check. An intermediate peer receives a request containing the source, the predecessor and successor peers. The intermediate peer checks and accounts the balance of the predecessor and successor peers, with the effect that the intermediate peer transfers its debts from the predecessor to the source. The intermediate peer sends a signed check with the new balance to the source peer. Each intermediate peer is requested sequentially to send a check to the requesting peer until the successor peer is the target peer. If an intermediate peer fails to send a check, *e.g.*, because of an imbalance due to old history data, then the path is invalid.

In Figure 3, peer s requests a resource with value 2 from peer t (1), assuming the initial state in Figure 2. Since the credit limit has been set to 1, peer t reports (2) that this exceeds the credit limit. Peer s evaluates its DTI and ITI tables. The ITI table of peer s contains the DTI from peer x (cf. Figure 2). The negative response for the request contains the DTI of peer t, so peer s updates its ITI accordingly in (2). Then, peer s searches a path using the ITI and DTI tables using breadth-first search and finds $s \to x \to w \to t$.

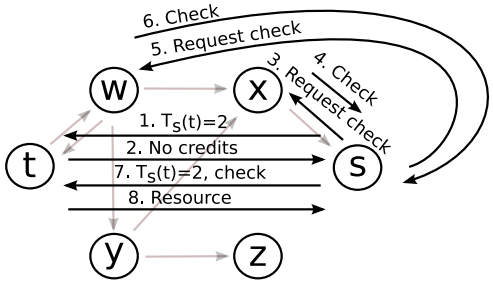

Fig. 3. Example: Peer s requests resources form peer t

In Figure 3 peer s requests from peer x (3) a check. Peer x settles and updates its DTIs for peer w and peer s to $DTI_x(s) = 0$ and $DTI_x(w) = 0$. The check that is sent back (4) contains a signed message with $ST_w(s) = 1$, where ST stands for settled transaction. Peer s contacts peer w (5) with this check and asks peer w to settle and update its DTIs for peer t and peer x, which results in $DTI_w(x) = 0$ and $DTI_w(s) = 0$, respectively. Then a check $ST_t(s) = 1$ is sent back (6). Peer s requests resources from peer t (7) and provides the check from peer w. Peer t settles and updates its DTI to $DTI_t(s) = 1$ and $DTI_t(w) = 0$. The request for a resource with value 2 is granted (8). After this transaction, peer t updates its DTI table with $DTI_t(s) = -1$.

4 Implementation and Simulation

PSH has been implemented and tested using Java 1.6. Message communication between peers is asynchronous. For short messages UDP is used, otherwise TCP.

4.1 Implementation

The time decay function $f_{decay}(transaction)$ is implemented as a queue with up to 100 entries to keep memory usage low. This means that the DTI and ITI tables contain up to 100 transaction entries each. The oldest entry will be removed if this limit is exceeded. An entry contains credit amount and node address. The transferred subset of DTI values is limited to 6 as a compromise between message size and data propagation.

Two versions of PSH have been simulated and compared to TFT: PSH as described in Section 3, and PSH with a reduced number of message transfers (PSH_r), both with $L = 3$. The reduction has been achieved by sending a subset of the DTI only if a request failed. PSH_r does not retry to send the request after a failed transaction, while PSH retries up to 3 times. A retry in PSH can be successful if a path can be found and verified. A retry in TFT would always fail because a peer only updates its history after a successful transaction. In contrast, PSH may update its history with a check from another peer and a retry may be successful.

4.2 Simulation

All simulations have been performed with 100 peers that share resources. Every peer has always exactly one resource, and requests 10 times a randomly chosen resource. Each simulation is run 10 times; averaged results are displayed. The simulation is repeated with the number of unique resources in the system varying from 1 to 100. In a system with only one unique resource, every peer requests the same resource, thus the interest is symmetric. With 100 unique resources on 100 nodes, there is a high probability that interest is asymmetric.

The success ratio is defined as $s/(f + s)$, where s is the number of successful transactions and f is the number of failed transactions. As shown in Figure 4, the success ratio of PSH is always higher than TFT, particularly when the number of unique resources is around 32. The higher success ratio is due to the shared history data. PSH_r is at an intermediate position, performing better than TFT, but worse than PSH, since the algorithm does not retry transactions.

The message count represents the total number of messages per transaction sent through the network. The total message size shows the total amount of bytes sent over the network. The message count and size include resource request and response messages, and for PSH additionally check request and reply messages.

In Figure 5, the message count for TFT is constant at 2 messages per transaction, since a transaction consists of a request and a reply message. In PSH, besides resource requests, peers also exchange checks, therefore, the number of

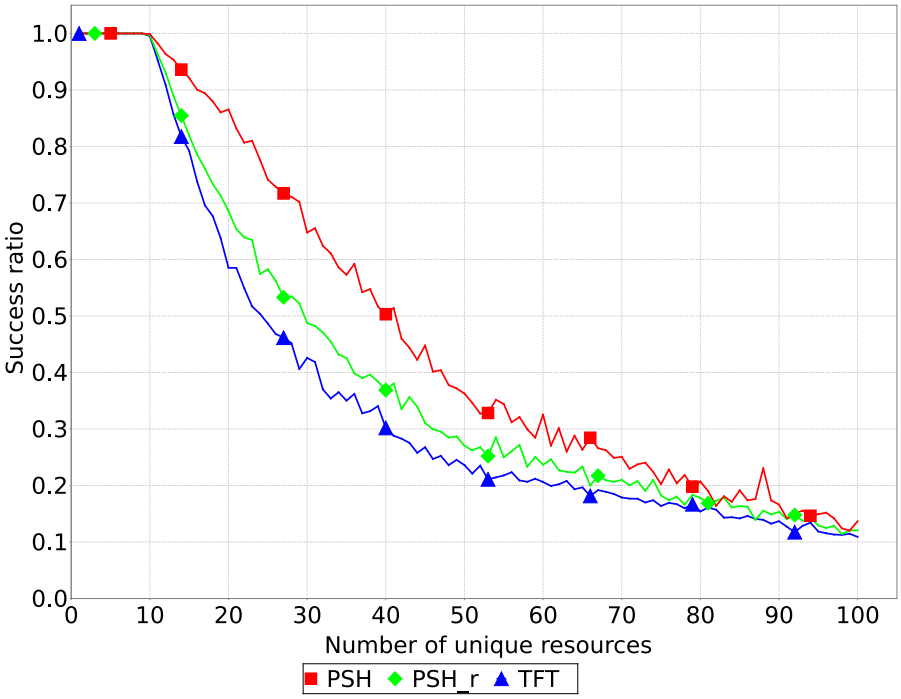

Fig. 4. Transaction success ratio for PSH, PSH(r), and TFT

messages is higher — up to twice as many as TFT, peaking on 32 unique resources. An important factor that contributes to this high number is the PSH retry behavior. Since PSH_r does not perform retries, its number of messages are, on average, only 2.22% above TFT, at maximum 6.4%.

In Figure 6, the message size is higher for PSH and PSH_r than for TFT due to two factors: (1) the message count is larger, and (2) messages contain also history information. The message size of PSH_r is smaller than PSH because of fewer history information.

Both Figures 5 and 6 show a similar behavior for PSH. First message size and count increases. This is due to the fact that the number of unique resources decreases. This leads to a decrease of the success ratio, which leads to message retries, with up to 3 retries. At the peak level, for about 33 unique resources (32 in our particular simulation), one resource is kept by approximately 3 peers. From that point on less than 3 peers have the same resource, thus, PSH sends fewer retry messages and the message size and count decreases.

5 Summary, Discussion, Conclusion, and Future Work

This paper presents PSH, a transitive TFT incentive mechanism, which has a higher transaction success ratio than TFT for peers with indirect reciprocity.

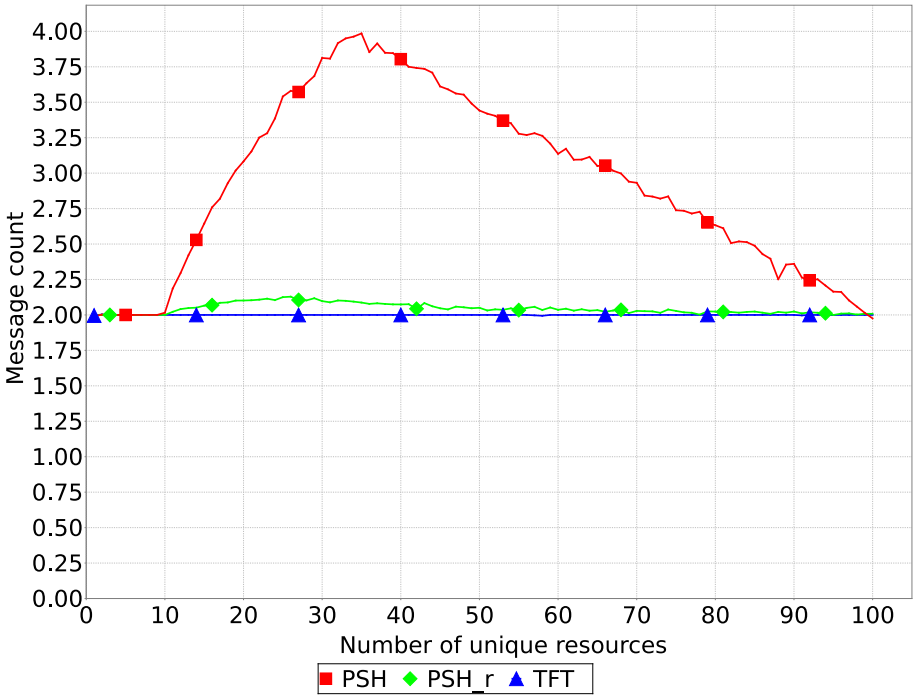

Fig. 5. Message count of PSH, PSH(r), and TFT per transaction

Simulations show the trade-off between a higher success ratio and a higher message size and count. PSH has a success ratio that is up to 73% higher than TFT. The message size is up to 5.6 times higher and the message count up to 2 times higher. However, with PSH modifications such as PSH_r, the success ratio can be increased, when compared to TFT, with only a small overhead with respect to message size (49% larger than TFT in average) and count (2.2% larger than TFT in average).

5.1 Discussion

A subset of history information is propagated together with request and response messages. Thus, if a peer does not interact with any other peer, history information does not flow and already stored information becomes inaccurate. A path that relies on inaccurate data will fail. In such cases, PSH retries the request, which result in a higher message size and count. To avoid this problem, a proper time decay function is applied.

A check from a peer has to be signed and thus, the public key has to be transferred first. Besides the message overhead, signing a message is a CPU intensive task. Thus, PSH is not suitable for micro-trading, i.e. trading many small resources in short time.

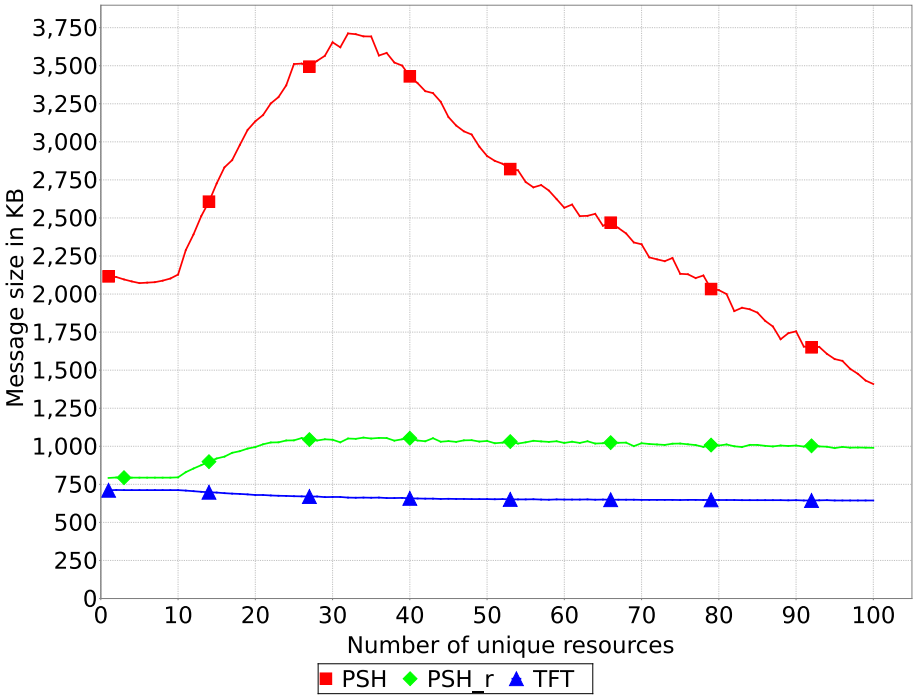

Fig. 6. Total message size of PSH, PSH(r), and TFT

5.2 Conclusion

PSH works better than TFT in systems with many different resources, i.e. in systems with an asymmetry of interest. This paper shows that there is potential to improve TFT by introducing PSH. The cost of a higher transaction success ratio is an increased message size. If a high transaction rate is more important than bandwidth constraints, then PSH should be preferred over TFT.

5.3 Future Work

The PSH mechanism could also be used for trust and reputation management. As the collected history information is locally available, the trust value can be calculated for each request and additional variables could be considered in the trust calculation.

The PSH mechanism has been thoroughly tested and simulated. While this first step of the evaluation shows that the mechanism works as expected, for the second step of the evaluation, further measurements in a distributed environment such as PlanetLab with more than 100 nodes are needed, in order to show that PSH is robust, fault and delay tolerant, and scalable. Future work will measure how PSH works with collusion and false reports, while varying the number of retries and the credit limit.

Acknowledgement

This work has been performed partially in the framework of the EU IST Project EC-GIN (FP6-2006-IST-045256) as well as of the EU IST NoE EMANICS (FP6-2004-IST-026854). The authors acknowledge valuable feedback from their colleagues and project partners.

References

1. Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *First Monday, Internet Journal*, 5(10), October 2000.
2. Farag Azzedin and Muthucumaru Maheswaran. Evolving and managing trust in grid computing systems. In *Canadian Conference on Electrical and Computer Engineering (IEEE CCECE)*, volume 3, pages 1424–1429, Winnipeg, Canada, May 2002.
3. Thomas Bocek, Mike Shann, David Hausheer, and Burkhard Stiller. Game Theoretical Analysis of Incentives for Large-scale, Fully Decentralized Collaboration Networks. In *Fifth International Workshop on Hot Topics in Peer-to-Peer Systems*, Miami, USA, April 2008.
4. Bram Cohen. Incentives Build Robustness in BitTorrent. Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.
5. Debora Donato, Mario Paniccia, Maddalena Selis, Carlos Castillo, Giovanni Cortese, and Stefano Leonardi. New Metrics for Reputation Management in P2P Networks. Technical report, Banff, Alberta, Canada, May 2007.
6. John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.
7. Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM Press.
8. Garrett Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, December 1968.
9. David Hausheer. *PeerMart: Secure Decentralized Pricing and Accounting for Peer-to-Peer Systems*. Number 16200. Shaker Verlag, Aachen, Germany, March 2006. ISBN 3-8322-4969-9.
10. David Hausheer and Burkhard Stiller. PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications. In *IFIP Networking Conference*, pages 40–52, Ontario, Canada, May 2005.
11. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.
12. Qiao Lian, Yu Peng, Mao Yang, Zheng Zhang, Yafei Dai, and Xiaoming Li. Robust Incentives via Multi-level Tit-for-tat. In *5th Int. Workshop on Peerto-Peer Systems (IPTPS)*, Santa Barbara, CA, USA, February 2006.

13. Richard T. B. Ma, Sam C. M. Lee, John C. S. Lui, and David K. Y. Yau. A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks. In *Joint international conference on Measurement and modeling of computer systems (SIGMETRICS '04/Performance '04)*, pages 189–198, New York, NY, USA, 2004. ACM.
14. Sergio Marti and Hector G. Molina. Limited Reputation Sharing in P2P Systems. In *5th ACM Conference on Electronic Commerce (EC '04)*, pages 91–101, New York, NY, USA, 2004. ACM Press.
15. Seth J. Nielson, Scott Crosby, and Dan S. Wallach. A Taxonomy of Rational Attacks. In *The 4th Annual International Workshop on Peer-To-Peer Systems (IPTPS 2005)*, Ithaca, NY, USA, February 2005. Springer Berlin / Heidelberg.
16. Philipp Obreiter and Jens Nimis. A Taxonomy of Incentive Patterns - The Design Space of Incentives for Cooperation. In *Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC'03)*, Melbourne, Australia, July 2003. Springer LNCS 2872.
17. Jeffrey Shneidman and David C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, February 2003.
18. Dan S. Wallach Tsuen-Wan Ngan and Peter Druschel. Enforcing Fair Sharing of Peer-to-Peer Resources. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, California, February 2003.
19. V. Vishnumurthy, S. Chandrakumar, and E. Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.
20. Li Xiong and Ling Liu. Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):843–857, 2004.
21. Beverly Yang and Hector Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 300–310, New York, NY, USA, 2003. ACM.
22. Mao Yang, Hua Chen, Ben Y. Zhao, Yafei Dai, and Zhengyou Zhang. Deployment of a large-scale peer-to-peer social network. Boston, MA, USA, June 2004.