Title
Bucking Free-Riders: Distributed Accounting and Settlement in Peer-to-Peer Networks

Authors
Agrawal, Abhishek
Brown, Douglas
Ojha, Aditya
et al.

Publication Date
2003-06-24

Peer reviewed

Bucking Free-Riders: Distributed Accounting and Settlement in Peer-to-Peer Networks

Abhishek Agrawal, Douglas J. Brown, Aditya Ojha, and Stefan Savage *
Department of Computer Science & Engineering
University of California, San Diego, CA

1 Introduction

The practice of *free-riding* – consuming service without providing equivalent service in return – is a well-documented problem in contemporary peer-to-peer (P2P) networks. For example, recent measurement studies of the Napster and Gnutella file-sharing systems reveal that the majority of files are provided by only 7% of hosts, while the majority of requests are generated by the remaining 93% [8, 1]. This "tragedy of the commons" effect is driven by the combination of individual self-interest coupled with the absence of adequate incentives to encourage (or enforce) fair use. Unfortunately, concentrating service among a small number of hosts in this manner is not only unfair, but it also undermines the key advantages of service distribution, including scalability, availability and robustness under attack.

Existing approaches to the free-riding problem fall into two rough categories: *reputation systems* [4, 6, 7] and *commerce systems* [3]. The first approach relies on identifying greedy or misbehaving users after the fact (either automatically or manually) and then refusing service to hosts with "bad" reputations. In contrast, the second approach is predicated on economic mechanisms that require each user to "purchase" service on demand, using a virtual currency that is obtained as payment for providing service in turn. While it is premature to reach strong conclusions about the limitations of each method, to a first approximation, reputation systems appear better suited to managing actively greedy users in a reactive fashion, while commerce systems are a more natural approach for proactively implementing appropriate incentives among passively self-interested users. We believe this latter property represents a more pressing requirement for large-scale peer-to-peer systems and consequently it is the focus of this paper.

There are two key components in any commerce-based resource management system: an accounting mechanism to securely store the currency held by individual users and a settlement mechanism to fairly exchange currency for services. Since the integrity of these services is critical, the simplest implementation approach, exemplified by Horn et al., is to centralize these functions within a single trusted third-party [3]. However, such a solution is at odds with the goals of P2P networks, which are by their nature highly distributed. Instead, this paper offers an alternative design that distributes accounting and settlement functions widely, providing powerful incentives without depending on a centralized trusted third-party to mediate each transaction.

In the remainder of this paper we present our overall system model, assumptions and requirements, followed by an in-depth description of the distributed accounting and settlement protocols. We conclude with a discussion of the performance implications and an examination of open research issues in this design.

2 System Model

Our system model presumes a large, structured peer-to-peer file sharing network, in which each user of the system has associated with them a currency balance that is maintained in an account. On a file transfer, the data provider's account is credited with an amount that is debited from the consumer's account. A user cannot arbitrarily create wealth in the system, preventing it from consuming indefinitely without also providing data to the system.

The currency held by each peer is stored in an account managed by a set of unbiased peers called *accountants*. Since peers are not presumed to trust one another, files are exchanged for currency through a multi-phase settlement transaction involving both peers, their respective accountants, and an optional mutually-selected third-party *mediator*. Integrity is established probabilistically through the distributed nature of the system—an adversary must maintain a large conspiracy of peers before it can subvert the accounting or settlement mechanisms.

Before describing these protocols in more depth, we briefly discuss the assumptions and requirements that we have set. Among the key assumptions:

- *Unique identity*. We assume the existence of a secure bootstrapping service that can establish the

* {abhishek, dbrown, aojha, savage}@cs.ucsd.edu

uniqueness of a given user and thereby prevent Sybil-style attacks on the system [2].

- *Rational actors*. Unless otherwise noted, we assume that all peers are primarily concerned with maximizing their own benefit rather than penalizing other users.
- *Limited adversary capabilities*. An adversary may compromise a *limited* set of peers and such peers may act in a Byzantine fashion with respect to the protocols described here. However, we assume that an adversary cannot easily compromise a large fraction of the accountants for a given user.

Assuming this environment, our design is motivated by the the following requirements:

- *Currency conservation.* A user must not be able to create new currency on demand. We do not require a closed steady-state economy—currency may be created independently of settlement—but the process of wealth creation cannot be manipulated by individual users.
- *Persistent accounting state.* We require that the balance of a user's account be stored in sufficiently robust a fashion that the failure of a small set of nodes does not threaten the persistence of this state.
- *Settlement atomicity.* We require that the exchange of currency for files should have atomic semantics. That is, users should be protected from an adversary defaulting on a settlement transaction and obtaining currency without providing service in return (or vice versa).
- *Neutral third-party incentives.* We require that, apart from the two principal actors, all other participants should have no incentive to misbehave in any particular settlement transaction.

In the following two sections we describe distributed accounting and settlement protocols that are designed to meet these requirements.

3 Distributed Accounting

Since our design distributes accounting state (i.e. currency balances) among many hosts, this presents two challenges: selecting which hosts will act as accountants for a particular user and ensuring the persistence and consistency of currency balances as host membership in the P2P network changes.

3.1 Accountant Selection

Accountant selection exploits distributed hash table (DHT) functionality whereby each node in the network is associated with a unique value from some identifier space [9].

Consider a node n with identifier id_n and a parameterized hash function $h(.,.)$ that maps a key to a value in a uniformly distributed manner. $r_1, r_2, \ldots, r_k$ are preselected global constants used to parameterize the hash functions to reduce the likelihood that two nodes share a subset of accountants. If we wish to select k accountants $A^1, A^2, \ldots, A^k$ to perform accounting for node n, we first compute:

$$\begin{aligned} A^1_{val} &= h(id_n, r_1) \\ A^2_{val} &= h(A^1_{val}, r_2) \\ &\vdots \\ A^k_{val} &= h(A^{k-1}_{val}, r_k) \end{aligned}$$

Now we may use the DHT operation $get\_node(V)$ to returns the identifier of the actual node that is nearest to V in the identifier space. In this manner we determine the identifiers of the accountants for each node n:

$$\begin{aligned} A^1_{id} &= get\_node(A^1_{val}) \\ A^2_{id} &= get\_node(A^2_{val}) \\ &\vdots \\ A^k_{id} &= get\_node(A^k_{val}) \end{aligned}$$

Although the identifiers $A^1_{id}, A^2_{id}, \ldots, A^k_{id}$ may change as nodes join and leave the system, the values $A^1_{val}, A^2_{val}, \ldots, A^k_{val}$ are static and depend only on the identifier id_n. When a node joins the system for the first time, it determines the identifiers of its accountants and sends a request to each of them to initiate a new account. From this time, the system will maintain persistent accounting state for this node. Since $r_1 \ldots r_k$ and $get\_node(V)$ are well-known and deterministic, this same process may be employed by any node to determine the accountants for a particular user.

3.2 Consistency and Replacement

While the aforementioned selection algorithm ensures that there are always k identifiable accountants for each node in the system, there must be a means by which accounting state is maintained when accountants are replaced due to dynamic system membership. For this purpose, each accountant i will periodically route a message to the nodes identified by $get\_node(A^i_{val})$ and $get\_node(A^{(i+1) \mod k}_{val})$ according to a system-configured polling interval T_{poll}.

When an accountant node hears such a message from itself, it knows that it has not been replaced by a new node whose identifier is closer to A^i_{val}. If it does not hear such a message, however, this signals that it has been replaced and must purge the relevant accounting state. Similarly, if a node that was not previously an accountant for node n hears such a message,

it knows that it has now been assigned to this role. It then queries the other $k-1$ accountants and awaits a *quorum* indicating the current accounting state for node n that it will record and maintain.

This polling facility ensures that the system functions correctly in the face of accountant failure so long as all accountants for a given node n do not fail within the same polling period. However, it is important that T_{poll} be selected to be greater than the minimum transaction time in the system. Otherwise, it is possible for accountants to fail without being replaced during a single transaction – ultimately undermining the systems' ability to achieve the necessary quorum.

4 Settlement Protocols

The second key feature of our design is a settlement protocol that allows currency to be moved between two parties in exchange for the delivery of services (i.e. a file transfer). This is similar in spirit to previous research in the realm of atomic exchange protocols [10]. Because accounting state is distributed throughout the network, every transaction involves the participation of the accountants for each party. Using the Small Byzantine Quorum (SBQ) protocol [5], the settlement protocol can maintain the consistency and integrity of a user's account until more than $(k-1)/3$ of their accountants are subverted. [1] Furthermore, in order to ensure currency conservation, all currency exchange occurs between accountants: no other party is ever in possession of a node's currency.

In the presentation of settlement protocols that follows, we denote the actors as follows: (B)illy is a buyer that desires to obtain some data object R; (S)usan is a seller that possesses object R; $A_{\{B\}}$ is the set of Billy's accountants; $A_{\{S\}}$ is the set of Susan's accountants, and M is an optional mutually-selected mediator.

The first protocol we present assumes that each of these actors exhibit purely rational behavior: they will not violate the protocol unless they directly benefit in doing so. This assumption allows a weak atomicity guarantee since actors will only exploit race conditions that benefit them significantly. The second protocol provides a stronger atomicity guarantee, and some protection against irrational users, using a third-party mediator to manage the settlement transaction at the expense of additional overhead.

4.1 Unmediated Transactions

Prior to the initiation of the transaction, Billy and Susan locate each other, negotiate a price X, and select a transaction identifier t_{id} through an out-of-band channel. Billy then identifies himself to each member of

[1] The SBQ protocol requires a read quorum of size $2f+1$ and a write quorum of size $3f+1$ in order to tolerate f failures.

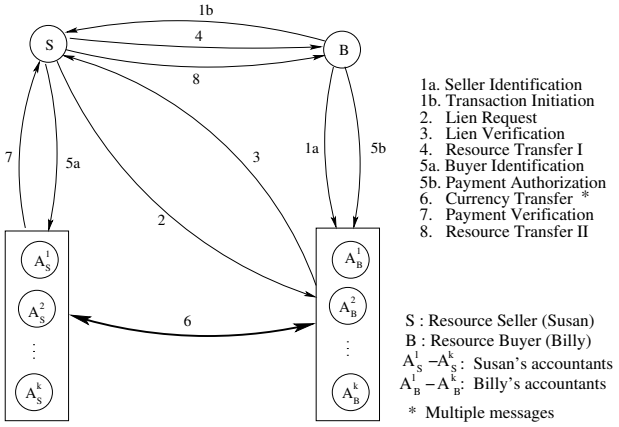

Figure 1: Unmediated Transactions

$A_{\{B\}}$ and indicates that they will be hearing from Susan regarding transaction t_{id}, and that he is willing to spend X currency units on this transaction that Susan may hold as a lien for up to duration T. This step prevents Susan from engaging in a denial-of-service (DoS) attack against Billy by placing a lien on his currency indefinitely or without his permission.

Billy next signals to Susan that he is initiating the transaction. At this point, Susan requests a lien from Billy's accountants to ensure that he will have sufficient currency to complete the transaction and to prevent him from engaging in a wealth manipulation attack. Upon hearing from a quorum of $A_{\{B\}}$ verifying this lien, Susan transmits a version of R encrypted with a symmetric key K_e of her choosing (denoted $E_{K_e}(R)$) to Billy. When she is finished transmitting $E_{K_e}(R)$, Susan also sends a message to each of her accountants identifying Billy and specifying the amount X of currency that it should expect to receive from $A_{\{B\}}$.

After receiving $E_{K_e}(R)$, Billy sends a message to each of his accountants authorizing it to complete the payment for the transaction. Because Billy has only received an encrypted version of the desired data, he cannot exploit the non-atomic exchange at this point, ensuring under the rational behavior assumption that there is no incentive for him to deny payment to Susan. This prompts $A_{\{B\}}$ to transfer X units of currency from Billy's account to Susan's and to guarantee that a quorum of $A_{\{S\}}$ receives a correct update from a quorum of $A_{\{B\}}$. This direct communication between the accountants ensures that Susan and Billy are unable to collude at this phase of the transaction.

After completing the currency transfer, each member of $A_{\{S\}}$ contacts Susan to indicate that her account has been credited. Assuming payment verification from a quorum of $A_{\{S\}}$, Susan completes the data transfer by transmitting K_e to Billy. As we assume that the act of sending the key K_e to Billy is

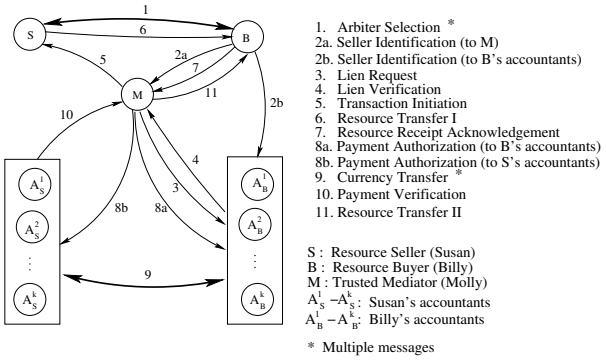

Figure 2: Mediated Transactions

of negligible cost, there is no incentive for a rational Susan to deny Billy this key.

4.2 Mediated Transactions

The preceding protocol allows for two unfamiliar parties to conduct a transaction while minimizing each actor's exposure to fraud. However, it is impossible to provide instantaneous atomicity in two party exchange – the nature of the activity dictates that there will be a point in such a transaction during which one irrational principal will be able to cheat the other by withholding the encryption key and prematurely terminating the transaction.

We now present an alternative protocol that uses a mutually-trusted third-party mediator in order to minimize the exposure to key withholding attacks. This mediator, (M)olly, is selected by Susan and Billy according to a process (given in Appendix A) whereby each selects half of the bits constituting its identify in order to ensure that he is unlikely to favor either primary actor.

As before, Billy and Susan establish a price X and a transaction identifier t_{id}. Billy then identifies himself to Molly and his accountants indicating his willingness to participate in transaction t_{id} with Susan. In this identification, Billy also indicates X and T (as in Section 4.1).

Should Molly be unwilling to participate in the transaction, it may abort at this stage, protecting itself from potential DoS by Billy and/or Susan. If it is willing to proceed, Molly next requests a lien from Billy's accountants and awaits its verification (as done by Susan in Section 4.1).

Assuming a correct verification from a quorum of $A_{\{B\}}$, Molly initiates the transaction by sending Susan a message confirming that he is willing to serve as mediator, that Billy is capable of proceeding with the transaction, and containing a randomly chosen encryption key K_A. This motivates Susan to randomly chose an encryption key K_B, encrypt R with K_B, and transmit $E_{K_B}(R)$ to Billy along with $E_{K_A}(K_B)$.

When $E_{K_B}(R)$ and $E_{K_A}(K_B)$ have been received, Billy acknowledges receipt of the encrypted data by sending $E_{K_A}(K_B)$ to Molly. Molly then decrypts this using K_A to obtain the key K_B and sends a message to each member of $A_{\{B\}}$ and each member of $A_{\{S\}}$ authorizing payment for the transaction.

This prompts $A_{\{B\}}$ to communicate with $A_{\{S\}}$ in such a manner as to transfer X units of currency from Billy to Susan (as in Section 4.1). After completing this currency transfer, each member of $A_{\{S\}}$ contacts Molly to indicate that the currency transfer has been completed. As in the previous protocol, the direct communication between the accountants for accounting purposes renders it impossible for Susan, Billy, and Molly to collude in any way to create currency in the system. Assuming payment verification from a quorum of $A_{\{S\}}$, Molly completes the data transfer by transmitting K_B to Billy.

5 Performance

The use of any of the above settlement protocols in combination with the distributed accounting scheme discussed in Section 3 provides a means of facilitating exchange in a distributed, non-hierarchical fashion. This is advantageous over a centralized approach in that it provides superior scalability, fault-tolerance, and resistance to attack. Nevertheless, the performance implications of our strategy merit discussion.

One metric relevant to performance is the bandwidth requirement imposed by our framework. Since all of the messages sent in the settlement protocols are small in size, particularly when compared to the size of the music and movie files typically shared on P2P networks—the bandwidth overhead of our protocols is not significant.

The SBQ protocol used to ensure accountant state consistency requires that the lien request, currency transfer, and payment verification steps in the protocols be conducted with a quorum of the accountants in agreement. As such, the number of messages exchanged for any of the protocols is dominated by the currency transfer phase, in which each accountant for S communicates with a quorum of the B's accountants to update S's account. Both Susan and Billy also issue $get\_node()$ requests to identify each others' accountants. Assuming k accountants for each node and a Chord-like $O(logN)$ DHT routing substrate [9], an $O(k^2 + k * logN)$ bound is implied on the total number of messages per transaction. We argue, however, that the number of messages is not the best indicator of system performance as perceived by an user.

Firstly, the $get\_node()$ requests can be sent in parallel and as part of the pre-protocol stage, ensuring that when the principal nodes identify each other, they do not contribute to the latency overhead imposed by the settlement protocols. Secondly, since communication with a node's accountants can occur in parallel, the overall latency for any step involving communica-

tion with accountants is independent of the number of messages exchanged. The latency of each step instead depends on the time taken by the last accountant in a quorum to respond. Assuming that t_{slow} is the upper bound on the time taken to send a small message to a correct node with the reliable delivery guarantee, the overall latency overhead excluding the data transfer phase is bounded by $(c*t_{slow})$ where the value of c depends on the number of sequential rounds. For example, for the unmediated protocol (Figure 1), the seller identification and transaction initiation steps can take place in parallel. Similarly sender identification and payment authorization can occur in parallel. Since currency transfer phase involves a two way communication, we get $c = 8$ for unmediated transactions. A similar calculation for mediated transactions (Figure 2), taking into account that mediator selection involves 3 sequential rounds gives $c = 13$.

Since t_{slow} refers to one way communication between two nodes, it can be approximated by $(rtt/2)$, where rtt is the worst case round trip latency in the network. Hence, the overall latency overhead, excluding the data transfer time, is bounded by $(4 * rtt)$ for unmediated transactions and $(6.5 * rtt)$ for mediated transactions. Assuming that the download bandwidth available to a broadband user is 500kbps, it takes 64 seconds to download a 4MB music file. Further assuming that the worst case round-trip latency is about 300ms, the overall latency overhead is roughly 2 seconds, less than 4% of the actual data transmission time. A more detailed analysis of these and other performance implications, like effect of accountant replacement strategy, remains a topic for future exploration. In particular, implementation of the proposed infrastructure would allow a quantitative assessment of our proposal.

6 Conclusion

The problem of free-riding in P2P systems is virtually ubiquitous. We have proposed in this paper a potential design for the technical infrastructure necessary to implement a sharing policy centered around micropayments. A scheme centered around a distributed set of accountants was suggested to provide a means of performing distributed accounting for each node in the system. Several settlement protocols were also presented that leverage this accounting facility. The relative merits and demerits of each of these schemes was discussed.

While our settlement protocols strive to ensure that users exhibiting rational behavior cannot benefit through improper behavior, there remains the reality that some users may engage in malicious, irrational behavior that could place other participants at a disadvantage. To help protect users from these situations, it could prove beneficial to implement a reputation system [4, 6, 7] within the resource sharing framework. In such a system, users would have the ability to report a transaction that they claim has been soured: the state recording this report would be maintained by the accountants of each primary actor in the transaction. A malicious seller may also cheat by sending spurious data instead of that which the buyer requested. The implementation of a mechanism within mediated transactions to verify the transmitted data in order to protect against this attack remains another area for future research.

References

[1] E. Adar and B. Huberman. Free Riding on Gnutella. *First Monday*, October 2000.

[2] J. Douceur. The Sybil Attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems*, March 2002.

[3] B. Horne, B. Pinkas, and T. Sander. Escrow services and incentives in peer-to-peer networks. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, 2001.

[4] A. Josang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, June 2002.

[5] J.-P. Martin, L. Alvisi, and M. Dahlin. Small byzantine quorom systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2002.

[6] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12):45–48, December 2000.

[7] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of the Fifth International Conference on Autonomous Agents*, June 2001.

[8] S. Saroiu, P. Krishna Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of the Multimedia Computing and Networking (MMCN)*, January 2002.

[9] I. Stoica, R. Morris, M. Frans Kaashoek D. Karger, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[10] J. Su and J. D. Tygar. Building blocks for atomicity in electronic commerce. In *Proceedings of the Sixth USENIX Security Symposium*, 1996.

Appendix A - Arbiter Selection

1. Billy chooses a sequence of $n/2$ bits, A_b. Susan similarly chooses a sequence of $n/2$ bits, A_s;

2. Billy sends $y = h(A_b)$ to Susan. Upon receipt of y, Susan sends A_s to Billy;

3. After receiving A_s, Billy sends A_b to Susan. Susan then verifies that $y = h(A_b)$. If this is not true, Susan aborts the protocol, otherwise each transaction participant can now identify the mediator as $A = (A_b A_s)$.