Probabilistic Quorum-Based Accounting for Peer-to-Peer Systems

William Conner and Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, IL 61801

Abstract

Providing an effective accounting scheme for decentralized peer-to-peer systems is difficult without relying on a central server or peers with special trusted status. In this paper, we present Probabilistic Quorum-based Accounting (PQA) for peer-to-peer systems. In our approach, a peer receiving a request will retrieve resource usage accounting information about the requesting peer from a quorum of peers to determine whether or not to grant the request. Our solution is completely decentralized and does not require a special trusted status for any peers.

1. Introduction

Many distributed applications are designed to run on top of peer-to-peer (p2p) routing infrastructures, such as [1,2,3]. These applications range from distributed storage systems [4,5] to web caching [6,7] to media streaming [8,9].

Although p2p designs offer many benefits, such as reduced costs and scalability, they also offer new opportunities for misbehavior that disrupts the system with too many requests. A selfish peer might request more than its fair share of resources for its own benefit, while a malicious peer might simply want to perform a denial-of-service attack by flooding the system with a large number of requests for disruption. In a system with selfish and malicious peers present, a peer P_1 providing a resource can easily determine how many resources a requesting peer P_2 is consuming from P_1, but it is difficult for P_1 to determine how many resources P_2 is consuming from other nodes in the system. If P_2 were selfish or malicious, then this would allow P_2 to make many moderately sized requests at many different nodes to get more than its fair share or disrupt the system. If P_1 contacts other peers to collect resource usage information for P_2, then it also needs to consider the possibility that some of those peers might provide false information about P_2.

Given this vulnerability in p2p systems, it becomes clear that p2p systems need effective accounting systems where peers can determine the resource usage of requesting peers in order to enforce usage policies. Unlike client-server architectures where all of the accounting information can be maintained at a central server, p2p architectures will need an accounting system that is more robust and scalable.

In this paper, we argue that probabilistic quorum systems [10] can be effectively applied to the problem of implementing accounting mechanisms for p2p systems without relying on a central server or nodes with special trusted status. In our solution, a node will first query a quorum of peers to acquire the current resource usage account information for a requesting peer. Based on that information, the requesting peer will be granted its request only if it is not found to have violated the system's usage policy. The probabilistic character of such quorum systems allows us to tune parameters affecting the tradeoff between security and communication overhead in our protocol. Probabilistic quorum systems also allow us to mask arbitrary failures from quorum members (e.g., failed or compromised peers providing false information).

Although we use probabilistic quorum systems [10], which were initially developed for maintaining consistent copies of replicated data with high probability despite server failures, our key contribution is the application of such quorum systems to the specific problem of implementing a p2p accounting system.

In the next section, we present related work. Section 3 introduces our probabilistic quorum-based accounting protocol. A simulation-based performance evaluation follows in Section 4. Finally, we conclude and discuss our future work in the last section.

2. Related Work

Yumerefendi and Chase discuss many accountability issues in dependable distributed systems at a high level, including the suggestion that probabilistic approaches might be more practical than enforcing full

accountability [11]. Although they discuss many interesting issues, no concrete solution is proposed.

The PlanetLab Central (PLC) infrastructure service uses a central database to enforce global resource utilization for an overlay testbed [12,13]. Such a centralized design lacks scalability and robustness to failures.

SHARP is another infrastructure service developed for use with PlanetLab [14]. SHARP is a distributed resource management service where resources are acquired through the exchange of resource claims. Unlike our approach (which can be tuned to tolerate the desired number of peer failures), in SHARP, if any single peer along a resource exchange path fails, then a new path will have to be discovered. This could significantly increase latency. Similarly, Scrivener also requires "credit path" discovery to prevent free-riding in content distribution networks [20].

Token-based accounting is a p2p accounting scheme based on exchanging tokens to access resources [15]. The issuance of new tokens in their scheme involves the cooperation of a strict quorum of *trusted* peers. Although token-based accounting also uses quorums, it uses strict quorums rather than using probabilistic quorums as we do. Also, unlike our approach, the quorums in token-based accounting are not designed to mask arbitrary failures among quorum members. The use of quorums in token-based accounting is limited to the process of exchanging foreign tokens for new tokens while our protocol is built around probabilistic quorum systems. Finally, token-based accounting requires a quorum of trusted peers while our probabilistic quorums can be selected among any peers in the system, which makes quorum selection easier.

Oversight is another p2p accounting scheme for p2p media streaming systems, which relies on a subset of trusted nodes [16].

3. Probabilistic Quorum-Based Accounting (PQA)

Probabilistic quorum-based accounting (PQA) is a p2p accounting system that relies on probabilistic quorums formed among regular nodes in the p2p system without requiring a central server or that any peers in the system have a special trusted status. This distinguishes our work from previous work. Accounting systems relying on central servers have a single point of failure, while systems relying on trusted nodes are vulnerable to the trusted nodes being targeted and compromised. To avoid these vulnerabilities, we chose a completely decentralized approach. Also, we chose probabilistic quorum systems rather than strict quorum systems to give us the flexibility to tune our quorum sizes to the desired level of security as explained in the following subsections.

The next subsection provides general background information on probabilistic quorum systems. After that, we will describe our p2p accounting system protocol.

3.1. Background

A strict quorum system is defined as a set of subsets of servers where every pair of subsets from the set intersects with each other [10]. As mentioned in [11], quorum systems can be used effectively to coordinate servers. Malkhi et al. introduce probabilistic quorum systems by relaxing the intersection property of strict quorum systems [10].

In probabilistic quorum systems, quorums are chosen according to some access strategy where each quorum will fail to intersect with some probability ε. Such probabilistic quorum systems are referred to as ε-intersecting quorum systems. Unlike strict quorum systems, our use of probabilistic quorum systems in our protocol allows us to adjust the size of the quorum to tune the security provided by the accounting system. Probabilistic quorums also simplify the access strategy for selecting peers compared to strict quorums. A more formal definition of probabilistic quorum systems, taken from [10], appears below.

> **Definition.** Let Q be a set system, let w be an access strategy for Q, and let $0 < \varepsilon < 1$ be given. The tuple (Q,w) is an ε-intersecting quorum system if $P(Q \cap Q' \neq \emptyset) \geq 1 - \varepsilon$, where the probability P is taken with respect to the strategy w.

Malkhi et al. also provide a construction of ε-intersecting quorum systems where the access strategy chooses a quorum of size $l\sqrt{n}$ uniformly at random from a universe of n servers where l is a constant that determines the size of ε [10]. Malkhi et al. also define (b, ε)-masking quorum systems, which are similar to ε-intersecting quorum systems, but every two quorums must have an intersection of at least size $2b + 1$ [10]. In (b, ε)-masking quorum systems, up to b arbitrary failures can be tolerated even if the data disseminated to the quorum is not self-verifying. Please refer to [10]

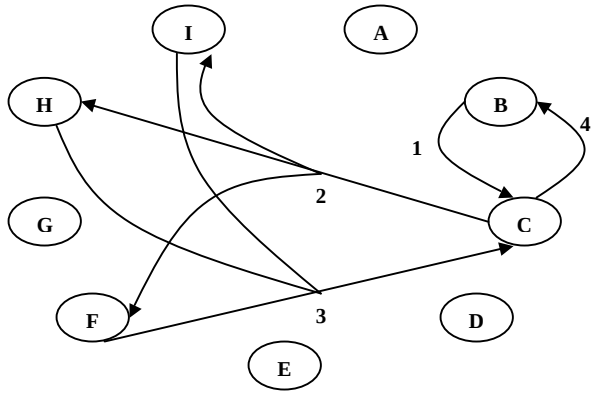
for more details and proofs concerning probabilistic quorum systems.

3.2. Protocol

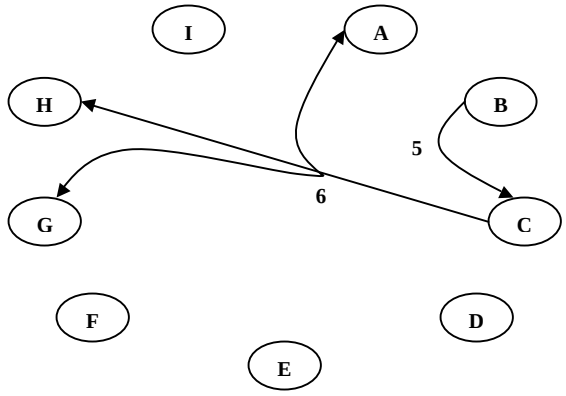
To provide an accounting system for p2p systems, we have chosen to use (b, ε)-masking quorum systems with a construction similar to the one presented in [10]. In our protocol, a peer P_1 desiring a resource or service located at peer P_2 will send a request to peer P_2. Assuming that each peer in the network has global membership information, P_2 will then initiate a *query phase* where it randomly selects a quorum of size $l\sqrt{n}$ and requests each quorum member's view of how much P_1 has consumed compared to how much P_1 has produced (i.e., P_1's accounting record). Each view is an accounting record containing the number of resource units consumed and the number of resource units provided by a particular peer. Of course, in general, peers that consume too much without providing anything in return should be denied future requests. The exact restrictions, however, are specific to the particular accounting system policy to be enforced. Our assumption about each peer having global membership information is reasonable in small overlay networks, such as the PlanetLab testbed [13]. However, larger scale networks would need a more complex strategy to find a quorum of size $l\sqrt{n}$ and developing an effective strategy to do this will be part of our future work.

Due to the intersection property of (b, ε)-masking quorum systems, we can determine the correct accounting record for P_1 based on replies to the query sent to the quorum even if b nodes in the quorum are malicious where malicious nodes falsify account information in their reply. Based on this information, P_2 will decide whether or not to grant the resource or service to P_1 depending on the specific accounting policy. If P_2 grants a request to P_1, then P_2 will collect a receipt signed with P_1's private key (i.e., the signature on these receipts can be verified with P_1's public key) during the *update phase*. The receipt should indicate that P_2 provided r units to P_1 where r is the number of units associated with that particular resource. If P_1 refuses to sign a receipt, then P_2 will refuse to serve future requests from P_1 and will report P_1 to a reputation system, such as those that appear in [18,19]. After collecting the receipt, P_2 will then disseminate copies of the receipt it receives from P_1 to another randomly chosen quorum of size $l\sqrt{n}$. The members of this quorum will update their local views on P_1's accounting record (of course, P_1 and P_2 will also update their respective views).

We will set l according to the level of security required for the particular p2p application and the number of peers expected to be malicious, which is equal to b. Since we have chosen (b, ε)-masking quorum systems, then even if b of the nodes in a quorum intersection are malicious, we still have enough nodes correctly following the protocol to determine the correct account information. Of course, larger values of the parameter l (i.e., larger quorum sizes) will increase the probability that any two quorums intersect, which increases the security of the accounting system because more nodes will have correct views of various peers' accounting records. However, there is a tradeoff in that larger values of l also mean that more messages will be sent per request since quorum sizes are larger. Below in Figure 1, we illustrate our PQA protocol with an example. Our example has the parameters l = 1 and n = 9 (i.e., quorum size is 3 for a network of 9 nodes).



(a) Query Phase



(b) Update Phase

Figure 1. PQA Protocol

Protocol Steps

1. Peer *B* requests a resource from peer *C*.
2. *C* queries a randomly selected quorum for views of *B*'s account record.
3. Quorum responds to *C* with *B*'s account information
4. *C* grants or denies *B*'s request
5. If request is granted, *B* sends receipt to *C*
6. *C* disseminates signed receipt from *B* to another randomly chosen quorum

Due to space limitations, we cannot include the analysis from [10] that derives the exact error probability ε that two (*b*, ε)-masking quorums do not intersect given the necessary parameters. However, our simulations presented in the next section provide some intuition on how increasing the quorum size also decreases the error probability ε, which leads to malicious nodes being granted fewer requests on average since larger quorums are more likely to intersect and are thus more likely to have up-to-date accounting information.

4. Performance Evaluation

We evaluated our solution by simulating a p2p network where nodes upload and download objects from each other. Each requested object in our simulation was worth exactly 1 unit with the account of the uploader being credited 1 unit while the account of the downloader is debited 1 unit. The policy to be enforced in our p2p network application simulation was that a peer should not be able to download greater than 5 units more than the number of units it has uploaded. For example, if a peer has only uploaded 5 objects, then that peer should not be able to download more than 10 objects in return. Such a policy could be used in a real p2p application to prevent free-riding and denial-of-service attacks.

Each simulation run had $n = 900$ nodes with a certain fraction of those nodes behaving maliciously. Malicious nodes in our simulation would each make 100 download requests to random benign nodes without uploading any requests in return (i.e., they only download without uploading). Also, in our simulation, whenever a malicious node was chosen to be a member of a quorum, the malicious node would provide false information indicating that the potential downloader was not violating the system's policy even if the potential downloader actually was in violation.

As far as the well-behaved nodes in our simulation, benign nodes faithfully followed the PQA protocol as specified in the previous section without attempting to violate the system's accounting policy. In our results below, we focus only on the effectiveness of our protocol at preventing malicious nodes from violating the system policy and the communication overhead.

For our simulations, we measured the average percentage of requests granted to malicious nodes as we increase the size of the quorums used (remember, we can increase the size of the quorum by increasing the parameter l mentioned in Section 3). More specifically, we set $l = 1,\ldots,20$ for different simulation runs, which resulted in the different quorum sizes that appear in Figure 2 below. We also considered two different scenarios in our simulations. The first scenario had $b = 9$ malicious nodes while the second scenario had $b = 45$ malicious nodes.

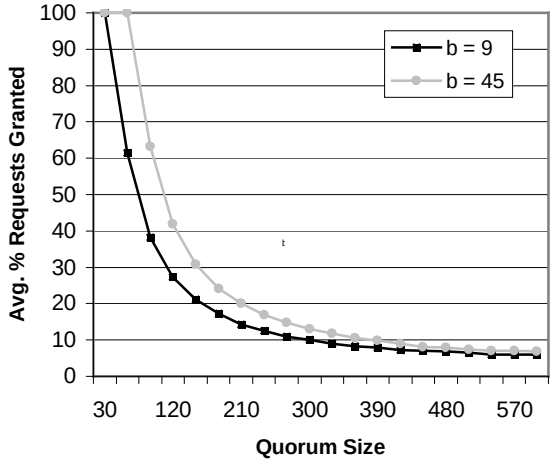


Figure 2. Average percentage of requests granted to malicious nodes as quorum size increases

As expected, our results demonstrate the tradeoff between security and communication overhead when using the PQA protocol for a p2p accounting system. For smaller quorum sizes, we send fewer messages per download request, which indicates less communication overhead. However, smaller quorum sizes also allow malicious nodes to download a larger fraction of the objects that they request. On the other hand, larger quorum sizes lead to more security since larger quorums restrict the number of download requests granted to malicious nodes more effectively. However, larger quorum sizes also lead to more communication overhead since more messages will have to be sent per download request. It should be noted that malicious

nodes will receive at least 5% of their requests due to the accounting policy used during our simulations. Of course, our graphs also show that networks with fewer malicious nodes can use smaller quorums to achieve the same limit on the percentage of requests granted to malicious nodes.

5. Conclusion and Future Work

Unlike previous approaches that rely on central servers or special peers with trusted status, probabilistic quorum-based accounting offers a flexible, effective solution to the problem of preventing selfish or malicious peers from consuming too many resources in a p2p system. Our decentralized approach is based on the application of probabilistic quorum systems to this problem, which relaxes some of the quorum intersection requirements of strict quorum systems to allow more flexibility.

Our future work will include a quorum selection strategy for p2p networks that does not have global membership information. Also, we plan to complete a full implementation of our accounting system for real experiments on PlanetLab in the future.

References

[1] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. "Chord: a scalable peer-to-peer lookup service for internet applications." *ACM SIGCOMM*, 2001.

[2] A. Rowstron and P. Druschel. "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems." *Middleware*, 2001.

[3] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. "Tapestry: a resilient global-scale overlay for service deployment." *IEEE Journal on Selected Areas in Communications 22(1): 41-53*, January 2004.

[4] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. "Wide-area cooperative storage with CFS." *18th ACM Symposium on Operating Systems Principles*, 2001.

[5] L. Cox and B. Noble. "Samsara: honor among thieves in peer-to-peer storage." *19th ACM Symposium on Operating Systems Principles*, 2003.

[6] P. Linga, I. Gupta, and K. Birman. "A churn-resistant peer-to-peer web caching system." *ACM Workshop on Survivable and Self-Regenerative Systems*, 2003.

[7] S. Iyer, A. Rowstron, and P. Druschel. "SQUIRREL: a decentralized, peer-to-peer web cache." *21st ACM Symposium on Principles of Distributed Computing*, 2002.

[8] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. "PROMISE: peer-to-peer media streaming using CollectCast." *11th ACM Conference on Multimedia*, 2003.

[9] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. "Distributing streaming media content using cooperative networking." *12th International Workshop on Network and Operating System Support for Digital Audio and Video*, 2002.

[10] D. Malki, M. Reiter, A. Wool, and R. Wright. "Probabilistic quorum systems." *Information and Computation Journal 170(2)*: 184-206, November 2001.

[11] A. Yumerefendi and J. Chase. "The role of accountability in dependable distributed systems." *1st Workshop on Hot Topics in System Dependability*, 2005.

[12] B. Chun and T. Spalink. "Slice creation and management." PlanetLab Design Note 03-013, July 2003.

[13] PlanetLab – Home. *http://www.planet-lab.org*.

[14] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. "SHARP: an architecture for secure resource peering." *19th ACM Symposium on Operating Systems Principles*, 2003.

[15] N. Liebau, V. Darlagiannis, A. Mauthe, and R. Steinmetz. "Token-based accounting for p2p-systems." *Kommunikation in Verteilten Systemen (KiVS)*, 2005

[16] W. Conner, K. Nahrstedt, and I. Gupta. "Preventing DoS attacks in peer-to-peer media streaming systems." *13th Annual Conference on Multimedia Computing and Networking*, 2006.

[17] M. Naor and U. Wieder. "Scalable and dynamic quorum systems." *22nd ACM Symposium on Principles of Distributed Computing*, 2003.

[18] E. Damiani, S. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. "A reputation-based approach for choosing reliable resources in peer-to-peer networks." *9th ACM Conference on Computer and Communications Security*, 2002.

[19] S. Kamvar, M. Schlosser, and H. Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks." *12th International World Wide Web Conference*, 2003.

[20] A. Nandi, T.-W. Ngan, A. Singh, P. Druschel, and D. Wallach. "Scrivener: providing incentives in cooperative content distribution systems." *Middleware*, 2005.