Enforcing Fair Sharing of Peer-to-Peer Resources

Tsuen-Wan “Johnny” Ngan, Dan S. Wallach and Peter Druschel
Department of Computer Science, Rice University

ProSeminar Peer-to-Peer Information Systems WS 04/05
Universität des Saarlandes
Speaker: Stefan Chouteau
25.01.2005

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

Overview

1. Introduction
2. Design Issues
3. Different Approaches of Implementation
4. Message Overhead Measurement
5. Conclusion

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

1. Introduction

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

1. Introduction

Problem:
In practice, Users have no natural incentive to provide services to their peers if it is not somehow required of them.

Solution:
Creating such requirements and implementing them directly into the peer-to-peer system

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

1. Introduction

Fundamental Thoughts:
How to require peers to provide resources to the p2p system?

Possibility:
Traditional Quota Enforcement Approach

But:
Traditional Quota Approach requires somekind of trusted authority to give a user permission to store files, but why should some users be placed in a position of authority over others in a system of peers?

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

2. Design Issues

2. Design Issues

Our Goals:

- Support a notion of fair sharing
- limit any node to only consume as much network storage as it provides space to others on its local disk
- Creation of a system of checks and audits directly into the peer-to-peer system

Enforcing Fair Sharing of Peer-to-Peer Resources

2. Design Issues

Threats to such a system:

- No Collusion:
  Nodes wish to gain an unfair advantage over the network, but they have no one to collude

- Minority Collusion:
  A subset of the p2p system is willing to form a conspiracy to lie about their resource usage. However, it is assumed that most nodes in the p2p network are uninterested in joining the conspiracy

- Minority bribery:
  The adversary may choose specific nodes to join the conspiracy, perhaps offering them a bribe in the form of unfairly increased resource usage

Enforcing Fair Sharing of Peer-to-Peer Resources

2. Design Issues

Incentive in such a System:

Think about the ability to consume resources as some kind of currency, where remote storage has more value to node than its local storage. When nodes exchange their local storage for other's remote storage, the trade benefits both parties, giving an incentive for them to cooperate. As such, there is no need for cash or other forms of money to exchange hands, the storge economy can be expressed strictly as barter economy.

3. Different Approaches of Implementation

3. Different Approaches of Implementation

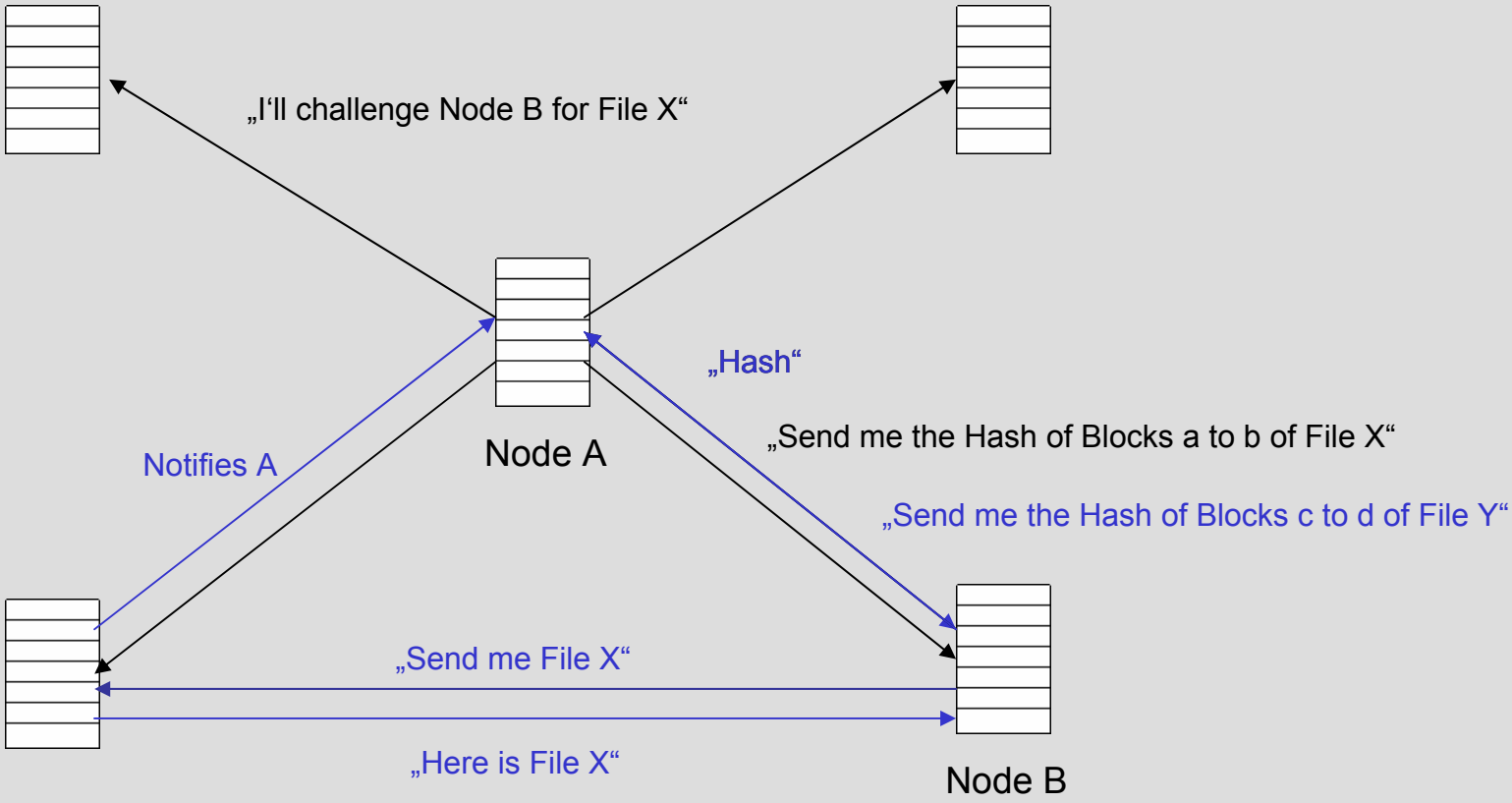
Assumption:

- existence of a public key Infrastructure, allowing nodes to digitally sign documents, such that any other node can verify them
- existence of a challenge mechanism to guarantee nodes are actually storing the files they claim to store

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

3. Different Approaches of Implementation – Challenge Mechanism



Enforcing Fair Sharing of Peer-to-Peer Resources

3.1 Different Approaches of Implementation – Smart Cards

Approach:

- Every Node in the System gets a Smart Card
- The Smart Card produces signed endorsements of other node's requests to consume remote storage

  [Space is charged to an internal Counter]

- Remote Storage can be reclaimed

  [When Storage is reclaimed, the remote node returns a signed message that the smart Card can verify before crediting its internal Counter]

3.1 Different Approaches of Implementation – Smart Cards

Pro:

- Smart Cards avoid the Bandwith Overheads of decentralized designs

Contra:

- Smart Cards must be issued by a trusted organisation
- Smart Cards must be periodically reissued to invalidate compromised cards
- Such a system would raise costs that have to be covered

→ **unsuitable for grassroots p2p systems**

3.2 Different Approaches of Implementation – Quota Managers

Approach:

- Same Design as Smart Card Approach but Smart Cards replaced by a collection of nodes in the p2p network
- Each manager must remember the amount of storage consumed by every node it manages and endorse all requests from the managed nodes to store new files
- Manager set for a node is a set of nodes adjacent to the node itself in the nodeID

3.2 Different Approaches of Implementation – Quota Managers

Drawback:

- Request approval causes high latency
- Number of malicious nodes must be less than one third of the manager set size *[vulnerable to Collusion / Bribery]*
- Managers suffer no direkt penalty if they grant requests that would be correctly denied

3.3 Different Approaches of Implementation – Auditing

Approach:

- Nodes are required to maintain and publish their own records
- Other nodes can audit this records

But:

Nodes have no inherent reason to publish their records accurately, because of that, we have to create *"natural economic disincentives"* to nodes lying in their records

3.3 Different Approaches of Implementation – Auditing

The Usage File:

Every node maintains a usage file, digitally signed, which is available for any other node to read. The usage file consists of three sections:

- The advertised capacity the node is providing to the system
- The local list of (nodeID, fileID) pairs, containing identifiers and sizes of all files the node is storing locally on behalf of other nodes
- The remote list of (fileID)s and sizes of all the files published by the node

3.3 Different Approaches of Implementation – Auditing

The Usage File:

The local and remote list describe all the credits and debits to a node's account.
We say a node is "under quota" and because of that allowed to write new files into the network, when its advertised capacity minus the sum of its remote list, charging for each replica is positive.

Example:

Advertised capacity of Client A: 17 MB

Remote list:

FileID	Filesize	No. Of replicas
(12)	3 MB	2
(3)	2 MB	1
(5)	100 kbyte	95

17.000 kbyte – (2 * 3.000 kbyte) – (1 * 2.000 kbyte) – (95 * 100 kbyte) = -500 kbyte

→ **Node is not under Quota, write access denied!**

3.3 Different Approaches of Implementation – Auditing

Cheating:

In this design, there are normally two possibilities to cheat on others:

- Inflate a node's advertised capacity

or

- deflate the sum of a node's remote list

This can be done by creating fraudulent entries in the remote or local list. To prevent fraudulent Entries we need an auditing procedure that a node may perform on another node to check the Lists for imbalances. They're called:

- **Normal Audit** *[check's a node's remote list]*

and

- **Random Audit** *[check's a node's local list]*

3.3 Different Approaches of Implementation – Auditing

Normal Audit:
For an entry in a node's own local list it checks if there is a corresponding entry in the appropriate node's remote list. If the entry is missing, the auditing node can feel free to delete the file.

Important:
Because an audit could be gamed if a node would know the identity of its auditor, anonymous communication is required.

3.3 Different Approaches of Implementation – Auditing

Random Audit:
For every entry in a node's local list the auditor checks if there is a corresponding entry in the appropriate node's remote list.

This would detect inconsitences in the audited node's usage file, but the node could collude with others. To fully audit the node, the auditor would need to audit the nodes in the audited nodes local list recursively. Implementing such a recursive audit would be prohibitively expensive.

Instead we require all nodes in the p2p overlay to perform random audits. Assuming all nodes perform these random audits on a regular schedule, every node will be audited with high Probability. The probability that a cheating node/anchor is not random audited by any node in one periode is about 0.368. A cheating anchor would be discovered in three periods with probability higher than 95%.

3.4 Different Approaches of Implementation – Extensions

Selling Overcapacity:

As described, a node cannot consume more resources than it provides to the network itself. But it is easy to imagine nodes that want to consume more resources than they provide or that provide more resources than they consume. This overcapacity could be sold, perhaps through a online bidding system for real-world money.
These Trades could be directly indicated in the local and remote lists, using entries like (NodeID , Amount Trade) for example, where the selling node writes the entry in its remote list and the buying node writes the entry in its local list.

Reducing Communication:

Another issue is that fetching usage files repeatedly could result in serious communication overhead. We could implement some improvements to reduce this overhead:

- sending the usage files directly through the internet, using an anonymizing relay
- Replica holders audit the publishing node alternately
- Only diffs of usage files are transmitted

4. Message Overhead Measurement

4. Message Overhead Measurement

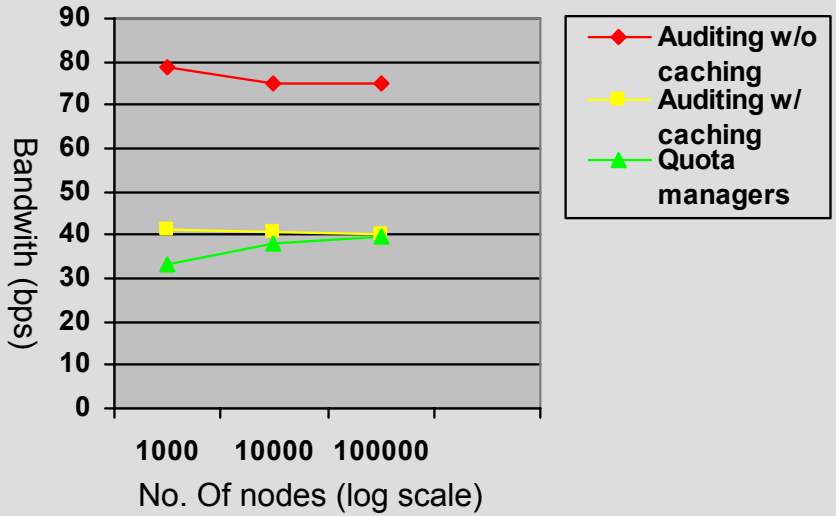
Experiment Details:

- For the simulation, we assume all nodes are following the rules and no nodes are cheating
- Storage space of each node is chosen from 2 to 200 GB with an average of 48 GB
- In each day of simulated time, 1% of the files are reclaimed and republished
- Two challenges are made to random replicas per file a node is storing per day
- For Quota Managers, the manager set size is ten
- For Auditing, normal audits are performed on average for times daily on each entry in a nodes remote list, random audits are done once per day
- Simulations have done with and without the append-only log optimization [diffs]
- Unless otherwise specified, all simulations are done with 10.000 Nodes, 285 files per node and an average node lifetime of 14 days

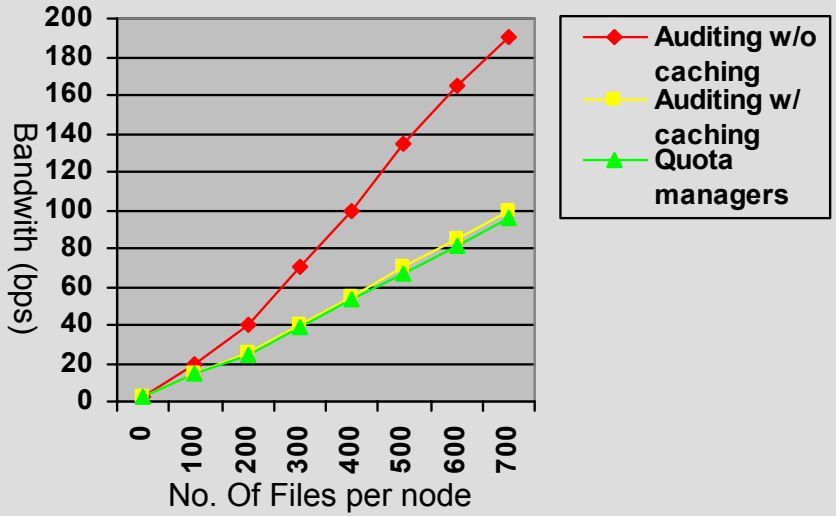
Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

4. Message Overhead Measurement

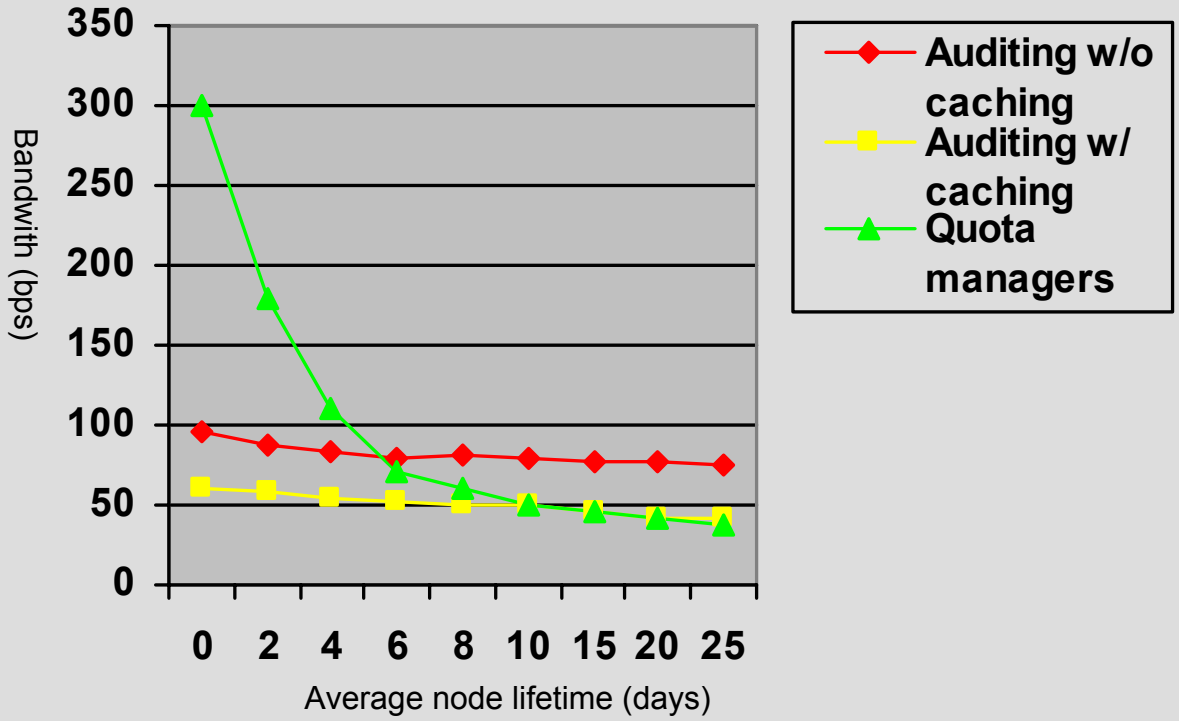


Overhead with different number of nodes



Overhead with different number of files per node

4. Message Overhead Measurement



Overhead with different average node lifetime

4. Message Overhead Measurement

Experiment Results:

The Experiment has shown that quota managers are mostly affected by the average node lifetime. The overhead for quota managers grows rapidly when the node lifetime gets shorter. That's, mainly, because of the costs in joining and leaving manager sets.

Auditing with caching has performance comparable to quota managers, but is not subject to bribery attacks and is less sensitive to the fraction of malicious nodes. Furthermore, in a variety of conditions, the auditing overhead is quite low – only a fraction of a typical p2p node's bandwith.

5. Conclusion

5. Conclusion

You've seen two architectures for achieving fair sharing of resources in p2p networks. Experimental results indicate small overheads and scalability to large numbers of files and nodes. In practice, auditing provides incentives, allowing us to benefit from its increased resistance to collusion and bribery attacks.

Enforcing Fair Sharing of Peer-to-Peer Resources

ProSeminar Peer-to-Peer Information Systems WS 04/05

Thank you for your attention.

The End