Differentiated Admission for Peer-to-Peer Systems: Incentivizing Peers to Contribute Their Resources

H.T. Kung¹ and C.H. Wu^{2,*}

¹Harvard University Division of Engineering and Applied Sciences Cambridge, Massachusetts, USA ²Academia Sinica Institute of Information Science Taipei, Taiwan, ROC

Workshop on Economics of Peer-to-Peer Systems June 6, 2003

Outline

• Introduction

Motivations and Main Ideas

- System Design
 - 1. Reputation-based, differentiated admission control
 - 2. Eigenvector-based reputation system
 - 3. Adaptation system of willingness-to-serve
 - 4. Sampling technique
 - 5. Distributed trust-enhancement scheme
- Summary and Concluding Remarks

A Challenge to P2P Systems: Fair Contributions of Resources by Participating Nodes



8-node transaction example

P₅ is likely a freeloader
because he *consumes*much more resources
than he *contributes*

 P_6 is very nice

Fairness is a well-known concern in P2P communities

Service Credit Matrix and Reputation Ranking: An Example



	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
P_1	0	0	1	1	0	0	0	0
P_2	0	0	1	1	2	0	0	0
P_3	0	0	0	1	3	0	0	0
P_4	0	0	0	0	4	0	0	0
P_5	2	0	0	0	0	0	0	0
P_6	0	0	0	2	4	0	0	0
P_7	0	0	2	1	3	0	0	0
P ₈	0	0	1	1	2	0	0	0

Service credit matrix

For illustration, we assume that a node will receive 2 credits for providing content and 1 credit for transporting content. Assume there are 9 transactions:

$P_{5} \leftarrow P_{6}$	$P_5 \leftarrow P_4$
$P_{5} \leftarrow P_{6} \leftarrow P_{7}$	$P_{5} \leftarrow P_{4} \leftarrow P_{3}$
$P_{5} \leftarrow P_{6} \leftarrow P_{7} \leftarrow P_{8}$	$\mathbf{P}_{5} \leftarrow \mathbf{P}_{4} \leftarrow \mathbf{P}_{3} \leftarrow \mathbf{P}_{2}$
$P_4 \leftarrow P_3 \leftarrow P_2 \leftarrow P_1 \leftarrow$	$-P_8 \leftarrow P_7 \leftarrow P_6$
$P_3 \leftarrow P_2 \leftarrow P_1 \leftarrow P_8 \leftarrow$	$-P_7$ and $P_1 \leftarrow P_5$

Service Reputation Usage Reputation

P ₁	7	4
P_2	5	4
P_3	4	3
P_4	2	2
P ₅	8	1
P_6	1	4
P ₇	3	4
P_8	5	4

Computed reputation rankings ₄

An Approach to Solving the Freeloader's Problem

Admission System: An Overview

When node X receives a request from node Y for content, X triggers a series of steps:

- 1. X grants the request with a probability based on X's current *willingness-to-serve* parameter ρ
- If granted, X determines whether Y should be "admitted" or "denied" by figuring out Y's service and usage reputation ranking from a set of sampling nodes
- If admitted, X sends Y the requested content and uses a third-party node to record X's credits for trust-enhancement purposes

Admission System: Main Ideas

- 1. A reputation-based, differentiated admission control that allows a node to receive a level of service based on its service and usage reputations in the past
- An eigenvector-based method that derives service and usage reputations of nodes by computing the largest eigenvalue / eigenvector pairs of the credit matrix associated with past transactions
- 3. An adaptation system that allows a node to take care of its own interest by contributing resources at a level just sufficient for its desired level of service
- 4. A sampling technique that uses top service and usage nodes, as well as benchmark nodes, to reduce the cost of computing service and usage reputations of nodes
- 5. A distributed trust-enhancement scheme that uses third-party nodes to manage and store credits required by reputation computations

Computing Reputations with Eigenvectors

Notations

S: service credit matrixvector s: service reputations $U = S^{T:}$ usage credit matrixvector u: usage reputations

An iterative method of computing s and *u*:

Node X's service reputation $s(x) = \sum_{x \in S} S(x, y)^* u(y)$

 $s^{(i+1)} = Su^{(i)}$ and $u^{(i)} = Us^{(i)}$

That is, $s^{(i+1)} = SUs^{(i)}$ or $s^{(i+1)} = SS^{T}s^{(i)}$

- Note that we can view the latter iteration as the power method of computing the largest eigenvalue/eigenvector pair of SS^T
- –This implies that s is the eigenvector of SS^T corresponding to the largest eigenvalue of SS^T. Similarly, u is the eigenvector of S^TS corresponding to the largest eigenvalue of S^TS

The matrix formulism here parallels to that used in ranking web pages (ref: Kleinberg HITS algorithm)

Reputation-based Admission

- Transactions \rightarrow Credit Matrices \rightarrow Reputations \rightarrow Rankings
 - After an allowed transaction is complete, the participating nodes will update their credits to reflect their roles in the transaction. Thus, the service and usage credit matrices will change accordingly. These changes will in turn affect future reputation rankings of the nodes
- The reputation ranking of nodes is comparative
 - We use s or u to denote both reputation and ranking depending on the context
 - Increasing the ranking of a node implies decreasing the rankings of some other nodes
- A request from node Y will be denied by node X if
 - Y's usage reputation **u** is above **A%** while its service reputation **s** is below **B%**
 - A and B are certain preconfigured thresholds with A>B (For example, A and B can be 80 and 20, respectively)
 - While being denied of receiving content, Y can still continue providing content and transport services, thereby improving its service reputation
- A freeloader which has a low service reputation (s < B%) and
 - a high usage reputation (u > A%) will be denied of service
 - The freeloader will need to either provide an increased level of service (to increase s) or reduce its content usage (to decrease u) in order to be readmitted again

Nodes' Adaptation in Willingness to Serve

Goal: search for a minimum level of contribution a node needs to provide in order to receive a desired level of service from other nodes



- willingness-to-serve ρ: a parameter determining the probability at which the node will grant arriving service requests
- desired service level C: a desired level of service. C is constrained by the system-wide parameters A and B: 1 - B*(1-A)
- **observed success rate** σ : the percentage of its content or transport requests that are granted by requested nodes

Intuitively, when a node finds that its success rate σ is above a desired service level C, it will decrease its willingness-toserve parameter ρ . On the other hand, when the node finds that its σ is below C, it will increase its ρ

Nodes' Finite State Machine for the Adaptation of Willingness to Serve



- A node increases its willingness-to-serve parameter ρ
- → This will improve its service reputation ranking s and cause some of the other nodes to drop their service reputation rankings or raise their usage reputation rankings
- → Some of these nodes may then enter the "Deny" state
- These nodes will then increase their ρ thereby allowing the current node to improve its success rate σ

"poor man's equilibrium" problem: if the two admit states were combined, many nodes could be in this combined "Admit" state with small σ and very low ρ . These nodes would never be able to increase their ρ since they cannot enter the "Deny" state due to their u being below A%

Newcomer Issue

For a new node, its initial state is configured to be "Admit & More" and its initial ρ is 0. That means it can receive services without providing services initially.



- If a new node makes many service requests such that its u is above A% and its s is below B%, it will enter the "Deny" state
- As long as its success rate σ does not reach its desired level of service C, it will stay at the "Admit & More" state and increase its ρ to raise σ
- When its σ reaches C, it will enter the "Admit & Not More" state and try to reduce its ρ

System Bootstrap

- Initially there have not been any transactions. Thus:
 - the service or usage credit matrix is initially a zero matrix
 - the success rate σ of every node is zero
- Before the success rate σ of a node reaches C, it will increase its ρ to raise σ
 - That means each node will try to obtain its desired level of service by providing more services

Convergence of Nodes' Adaptation: A Simulation Result





- When in the "Admit & Not More" state, a node decreases ρ using $\rho \leftarrow .95 * \rho$, else it increases ρ using $\rho \leftarrow max (\rho + .05, 1)$.Except when $\rho = 1$, the decreasing rate is smaller than the increasing rate
- The simulation results above show that
 - $-\sigma$ generally tracks changes in **C** as the node will adapt its ρ value
 - A new node will be able to adapt its ρ to achieve its desired level of service C within about 16 rounds

Sampling Heuristic: Use of Benchmark Nodes

To determine the current state of a node X, we will only compare X to two benchmark nodes:

Node P_A has its usage-reputation ranking percentile at A% and Node P_B has its service-reputation ranking percentile at B% Need not know the exact rankings of a node.



A Top-nodes Sampling Heuristic

In computing reputation, we will only use a sampling set, consisting of top service and usage nodes, node X, and two benchmark nodes P_A and P_B



Effectiveness of the Sampling Heuristic: A Simulation Result



When transactions exhibit the "hub" phenomenon, it generally suffices to use a small number of top nodes, such as 10, in the sampling set

Idea 5: Distributed trust-enhancement scheme

Distributed Trust-enhancement Using a Third-party Node



- For a transaction of sending content from P_i to P_i
 - The associated service and usage credits are stored at a third-party node P_h , where h = hash (i, j)
 - We can use a distributed hash table mechanism (DHT) such as Chord to maintain the credit matrix
- Generally we can assume that P_i and P_j have no management authority on data stored at third-party nodes

Summary and Concluding Remarks

- A distributed admission system with following features:
 - 1. reputation-based admission control Consider the service *and* usage reputation rankings of a requesting node as well as the willingness of the requested node
 - 2. service and usage reputations computed with eigenvectors Consider the authorities of individual nodes
 - 3. the 3-state model and nodes' adaptation on ρ Search for a minimum level of contribution a node needs to provide
 - 4. use of sampling with top nodes and benchmark nodes Reduce the information needed to determine current state of a node
 - 5. trust-enhancement with third-party nodes Provide protection against possible cheating of the system
- Preliminary simulation results suggest that the scheme may work
- Further research into control strategies on ρ and various application-specific scenarios would be useful