CSF641 – P2P Computing

點對點計算

Incentives in Peer-to-Peer Systems

吳俊興

國立高雄大學 資訊工程學系

Outline

I. Incentives and Free-riding Problems

II. Basic Incentive Techniques

- Payment-based approaches
- Reputation-based approaches

III. An example: Differentiated Admission

IV. Summary

Free-riding Problems

- Free-riding problem: there are many free loaders in an open P2P system
  - Free loader: downloading files while not contributing any to the network
  - a.k.a. Free-loader problem
- [2000] study on Gnutella 0.4:
  - 66% of peers shared nothing, 73% shared 10 or fewer files
  - Top 1% of sharing peers accounted for 47% of all QueryHits, and top 25% of these peers provided 98% of the QueryHits
- [2005] revised study on Gnutella 0.6:
  - 85% of peers sharing nothing, 86% share 10 or fewer files

Why Free-riding Problems Matter

"***Tragedy of the digital commons***": significant numbers of free riders degrade the entire system's utility

- Fairness issue: it happens often that a peer consumes much more resources than he contributes
- Efficiency issue: if there are fewer freeloaders
  - A provider can distribute contents quickly
  - A consumer can obtain contents quickly

If individuals gain no personal benefits from uploading files, it's ***rational*** for users to free ride

- In fact, it's inconvenient and costing
- Running P2P software has the risk of security threat
- NAT and firewall make free-riding worse

Incentives

- Definitions of Incentive
  - *Merriam-Webster Online Dictionary*: something that incites or has a tendency to incite to determination or action
  - *Cambridge Dictionary*: something which encourages a person to do something
- Solving free-riding problems with decentralized incentive techniques
  - It’s difficult to ensure that all autonomous peers follow the same policies or rules
  - It’s easy for any node to manufacture any number of identities

Outline

I. Incentives and Free-riding Problems

II. Basic Incentive Techniques

- Payment-based approaches
- Reputation-based approaches

III. An example: Differentiated Admission

IV. Summary

Incentive in eMule and KaZaa

- Credit system (eMule)
  - eMule rewards users for uploading files
  - Client credits are not global
  - Credit=min(Uploaded Total x 2 / Downloaded Total, SQRT(Uploaded Total + 2))
- Participation Level (KaZaa)
  - A function of download and upload volumes, and peers with high participation levels have higher priority
- Disadvantage
  - Relatively weak incentives for cooperation since free riders can still benefit from it, if they are patient enough to wait
  - Favor users with high access bandwidth

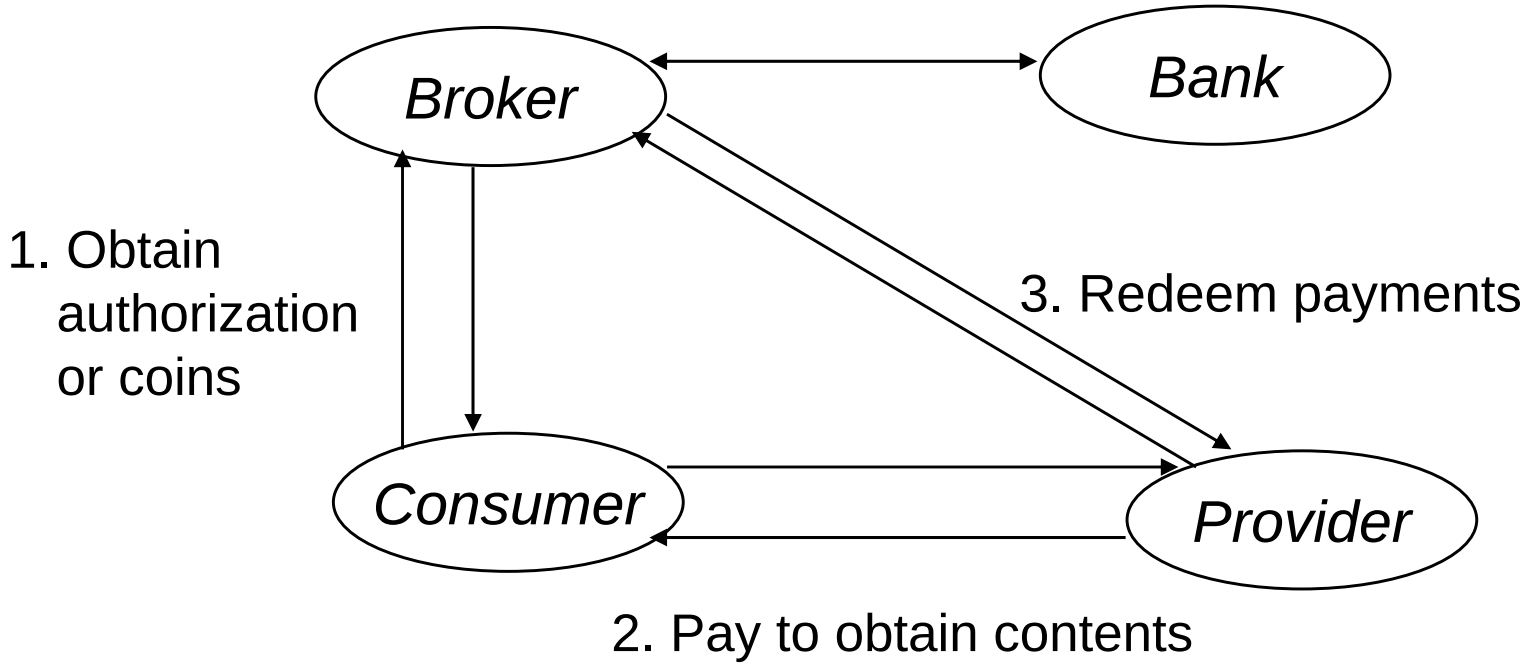
Incentive in Maze

1. New users are initialized with 4096 points.
2. Uploads: +1.5 points per MB uploaded
3. Downloads:
   - -1.0/MB downloaded within 100MB
   - -0.7/MB per additional MB between 100MB and 400MB
   - -0.4/MB between 400MB and 800MB
   - -0.1/MB per additional MB over 800MB
4. Download requests are ordered by $T = requestTime - 3\log P$, where P is a user's point total.
5. Users with $P < 512$ have a download bandwidth of 200Kb/s.

Related Works

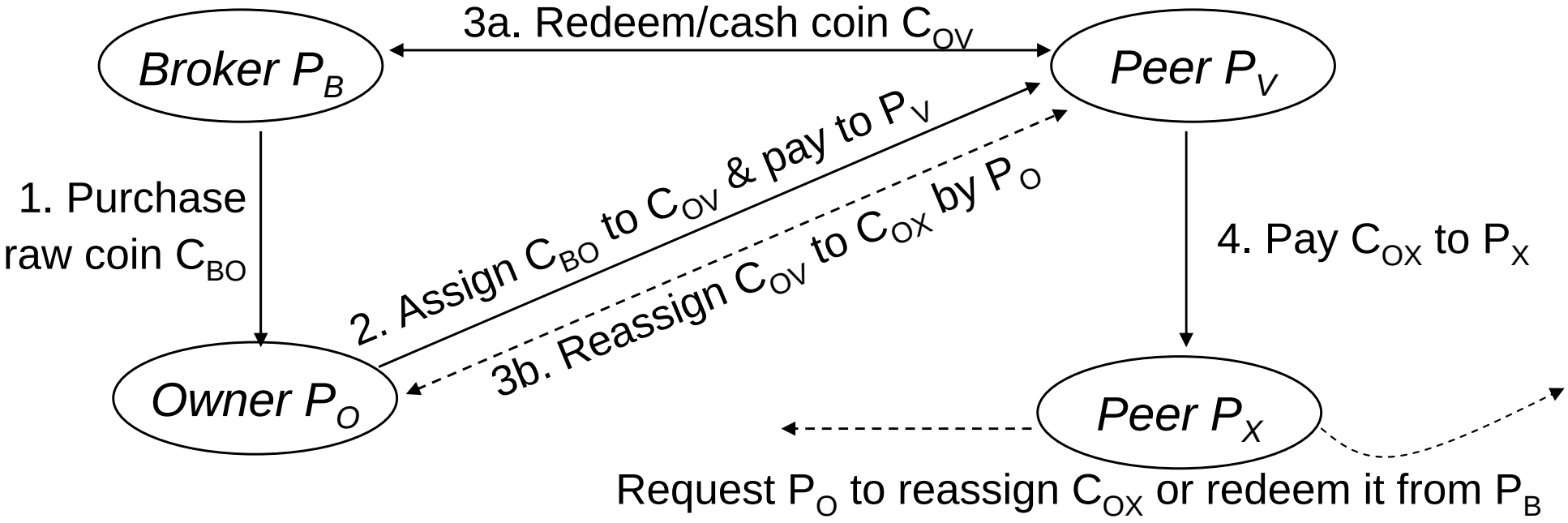
- Access Control List
  - White list and black list
- Payment-based Approaches
  - PPay, KAMAR, MojoNation
- Reputation-based Approaches
  - game-theoretic framework: BitTorrent
  - trust management: Reputella, NodeRank, PeerTrust, EigenTrust, Free Haven

Payment-based Approaches



- Online broker: heavy-loaded
  - message complexity = O(# of transactions)
- Offline broker: double spending and double payment
- Further issues: newcomer and privacy

PPay: Offline Broker



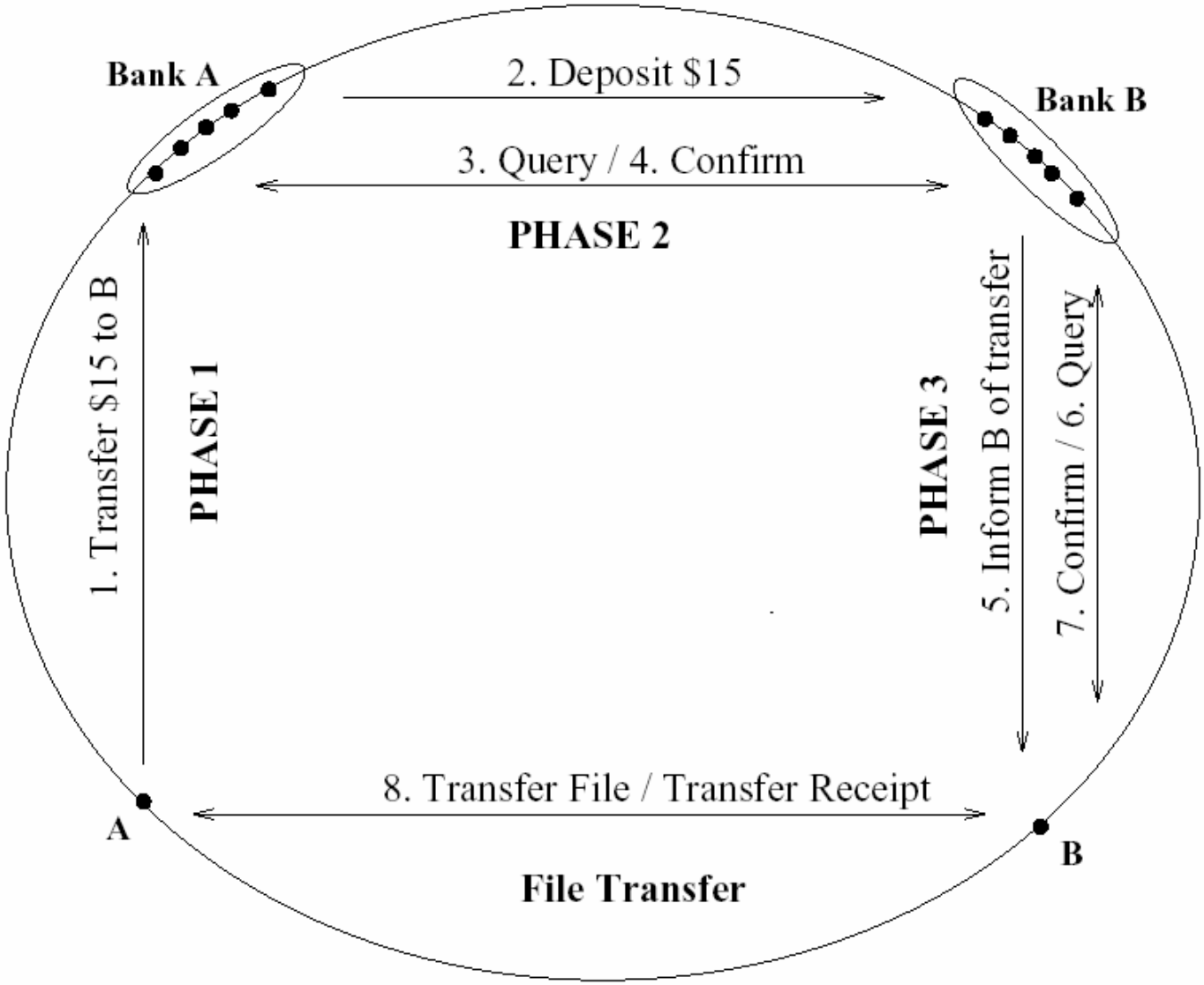
- *Owner* peer P_O purchases raw coin C_{BO} from broker P_B:
  - C_{BO}= {P_O, serial number sn} Signed by B
- P_O buys a service from peer P_V with the *assigned coin* C_{OV}
  - C_{OV} = {P_V, seq1, C_O} signed by P_O
- P_V: instead of cashing C_{OV} from broker P_B, P_V can request P_O to reassign C_{OV} to C_{OX}, to be paid to another peer P_X
  - Reassignment request from P_V to P_O: R_{OVX} = {X, C_{OV}} signed by P_V
  - ***reassigned coin***: C_{OX} = {X, seq2, C} signed by P_O

PPay: Offline Broker *(cont.)*

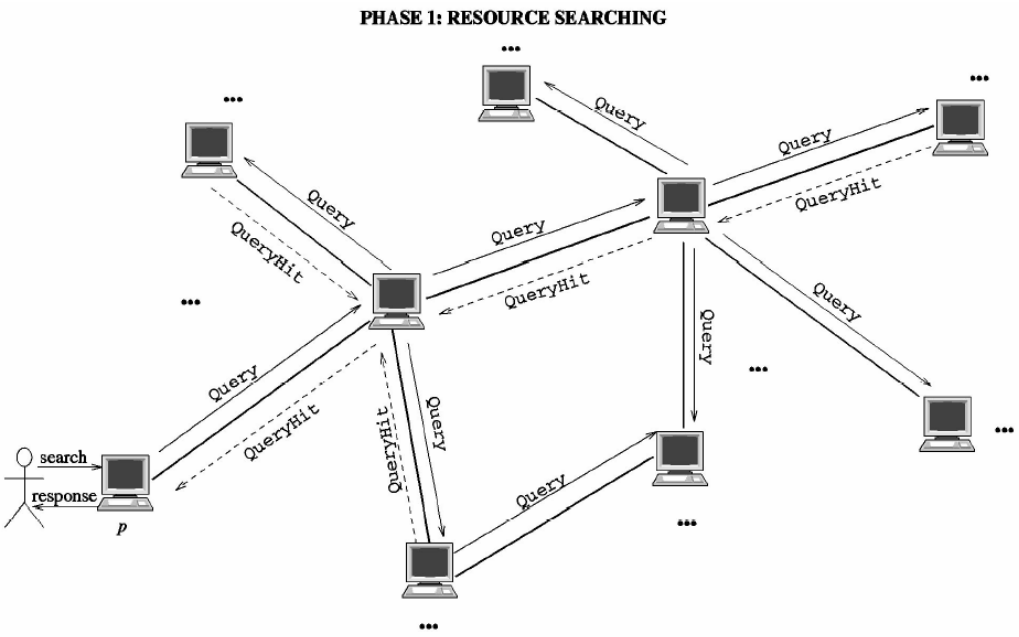
- Self-managed, floating currency to reduce broker involvement
  - Owner peer P_O is responsible for the purchased raw coin C_{BO}
    - Assign or reassign C_{BO} to C_{OX} for another peer P_X
  - Message complexity for broker P_B: O(# of coins issued)
- Does not prevent coin fraud, but ensure

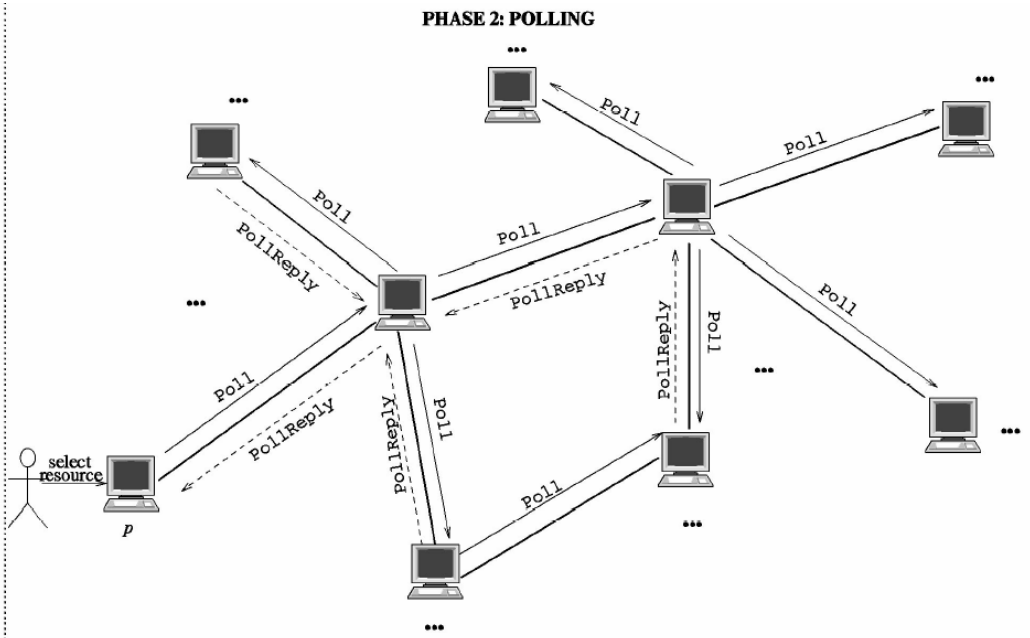
1. Any fraud can be detected and traced back to the misbehaving peer
   - Reassign once by P_O for coin C_{OV} by ascending *seq*
2. Proper punishments are in place
   - Broker P_B punishes P_O if P_O assigns or reassigns C_{BO} for multiple times
   - Detected by P_B if C_{BO} is redeemed by multiple peers with valid assigned or reassigned coins

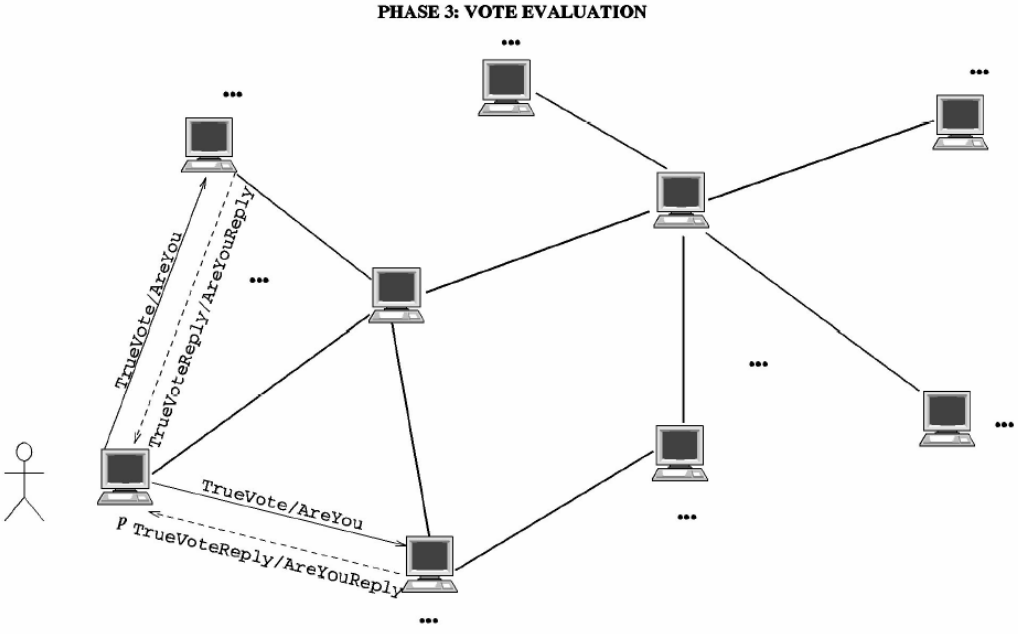
KARMA: Bank Set

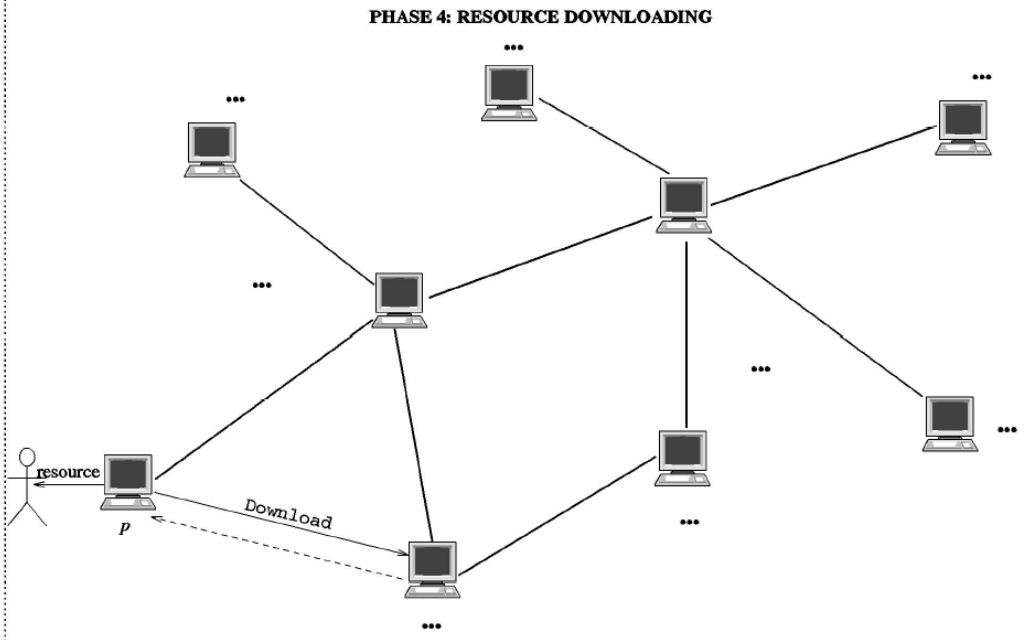


P2PRep/Reputella





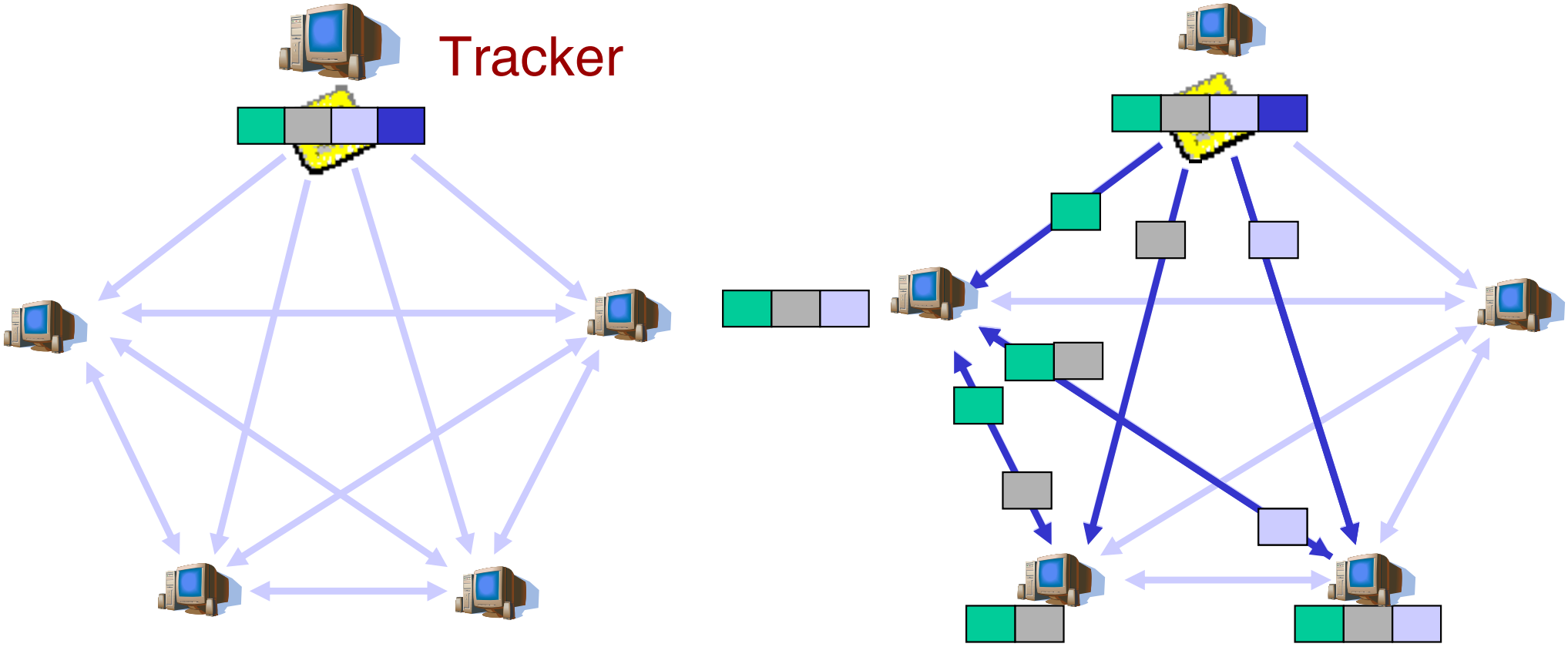




BitTorrent: Swarming and Tit-for-Tat

- Swarming: splitting and pipelining
  - **Search**: Out-of-band. E.g., use Google to find the .torrent metafile for the file you want
    - file information: length, name, hashing information
    - URL of tracker
  - **Join**: contact centralized “tracker” server, get a random list of peers
  - **Publish**: Run a tracker server
  - **Fetch**: Download chunks of the file from your peers. Upload chunks you have to them.
- Tit-for-tat: “I’ll share with you if you share with me”
  - Be optimistic: occasionally let freeloaders download

BitTorrent: Publish/Join and Fetch



Outline

I. Incentives and Free-riding Problems

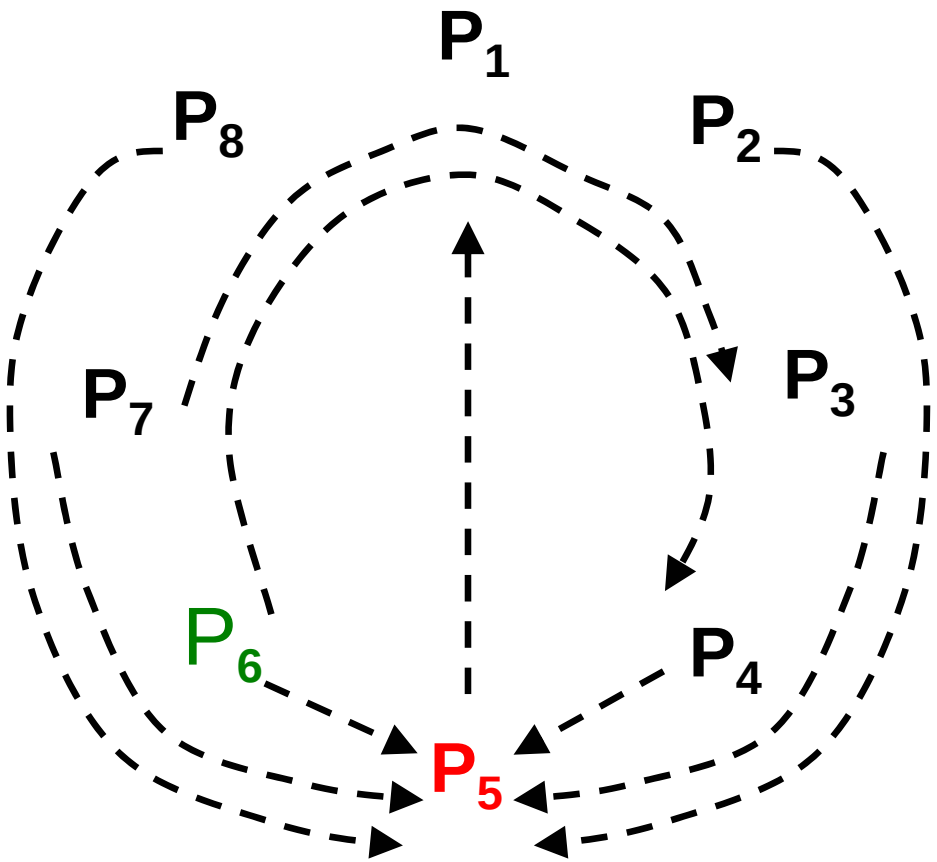
II. Basic Incentive Techniques

- Payment-based approaches
- Reputation-based approaches

III. An example: Differentiated Admission

IV. Summary

Fair Contributions of Resources by Participating Nodes



8-node transaction example

- P_5 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

Peer Selection and Peer Pressure

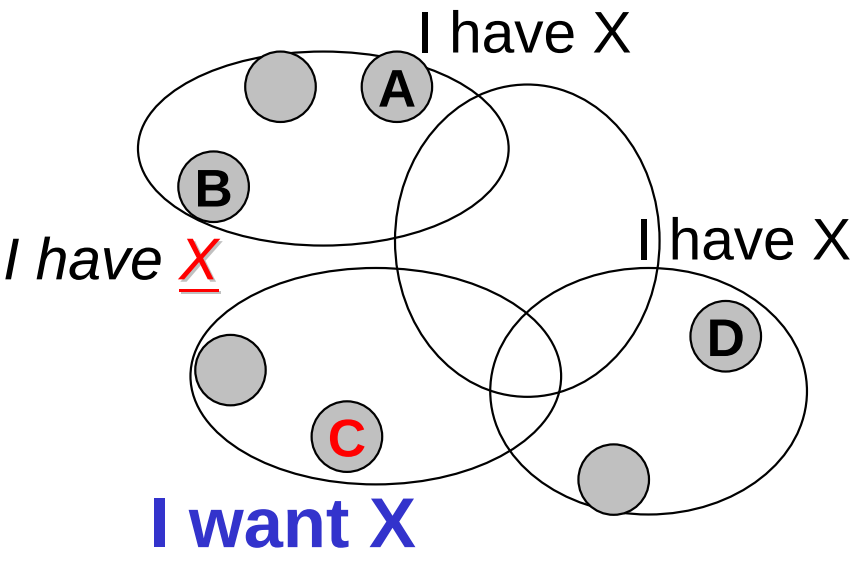
In an open P2P, it happens often that

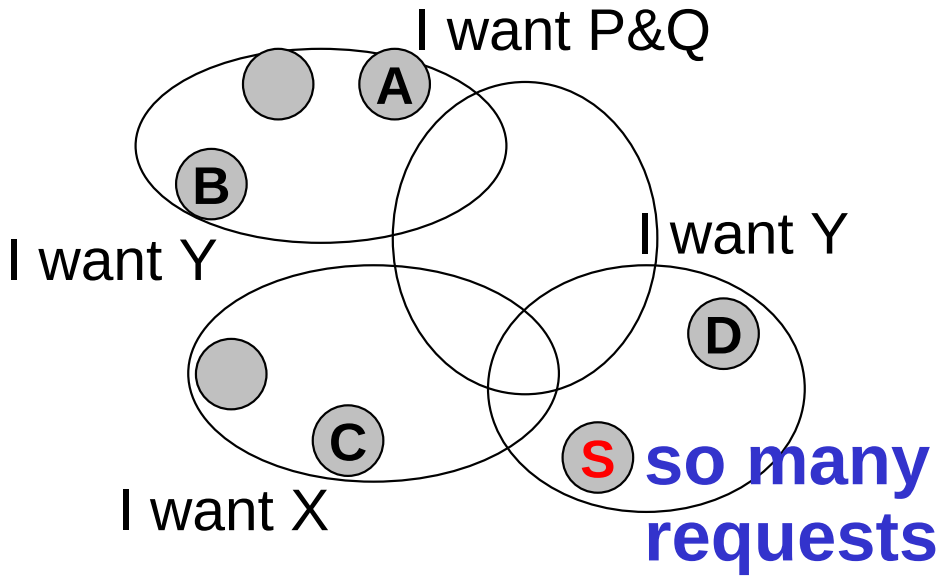
Server Selection

a requesting peer (client) needs to decide which servers it should request service from

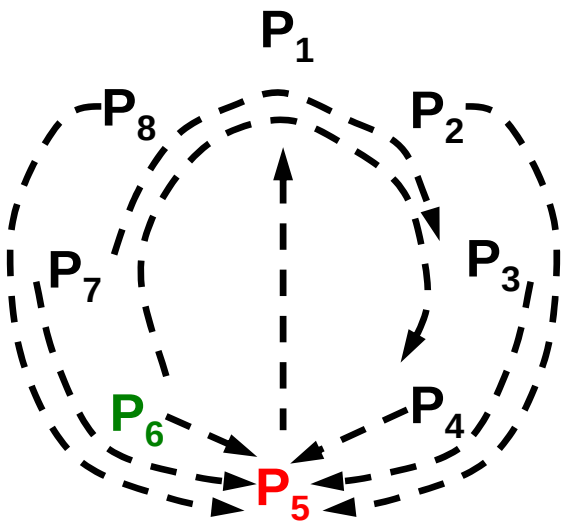
Client Selection

a supplying peer (server) needs to decide which clients it should grant requests first





PeerRank - Reputation Ranking



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:

$P_5 \leftarrow P_6$
$P_5 \leftarrow P_6 \leftarrow P_7$
$P_5 \leftarrow P_6 \leftarrow P_7 \leftarrow P_8$
$P_4 \leftarrow P_3 \leftarrow P_2 \leftarrow P_1 \leftarrow P_8 \leftarrow P_7 \leftarrow P_6$
$P_3 \leftarrow P_2 \leftarrow P_1 \leftarrow P_8 \leftarrow P_7$ and $P_1 \leftarrow P_5$
$P_5 \leftarrow P_4$
$P_5 \leftarrow P_4 \leftarrow P_3$
$P_5 \leftarrow P_4 \leftarrow P_3 \leftarrow P_2$

	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

	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

Admission System: An Overview

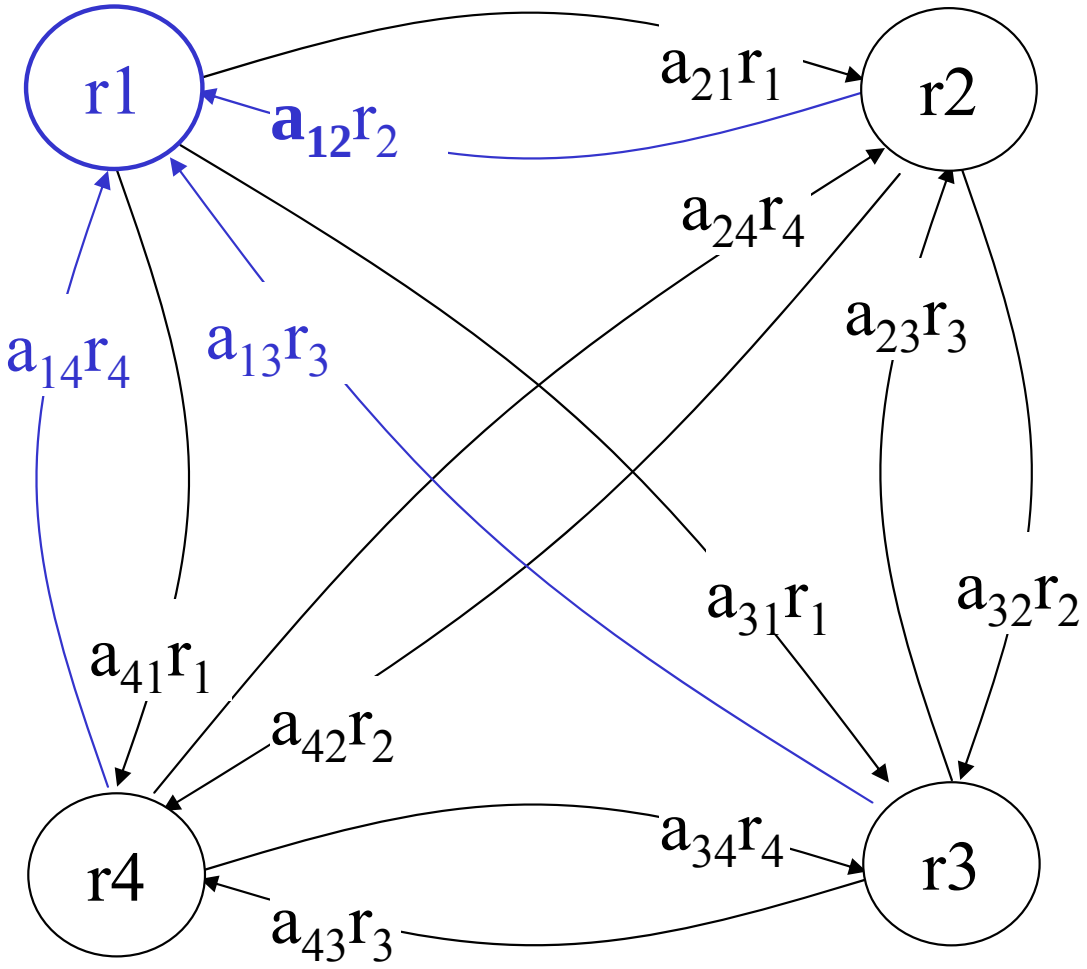
When node X receives a request from node Y for content, X triggers a series of steps:

1. X grants the request with a probability based on X's current *willingness-to-serve* parameter ρ
2. 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
3. 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
2. 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

Eigen-systems



$$\mathbf{A}x=\lambda x$$

x: eigen-vector

λ: eigen-value

$$\underset{r_{new}}{\begin{pmatrix} r_1' \\ r_2' \\ r_3' \\ r_4' \end{pmatrix}} = \underset{A}{\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}} \underset{r_{old}}{\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}}$$

$$r_1' = a_{11}r_1 + a_{12}r_2 + a_{13}r_3 + a_{14}r_4$$

$$x' = \mathbf{A}x$$

Computing Reputations with Eigenvectors

Notations

S: *service credit matrix* vector ***s***: *service reputations*

U = **S**^{T}: *usage credit matrix* vector ***u***: *usage reputations*

An iterative method of computing *s* and *u*:

Node X's service reputation

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

$u^{(i)} = Us^{(i)}$ That is,

$s^{(i+1)} = SUs^{(i)}$ or $s^{(i+1)} = SS^Ts^{(i)}$

$$s(x) = \sum_{y=1}^{n} S(x,y) * u(y)$$

$$u(x) = \sum_{y=1}^{n} U(x,y) * s(y)$$

– 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

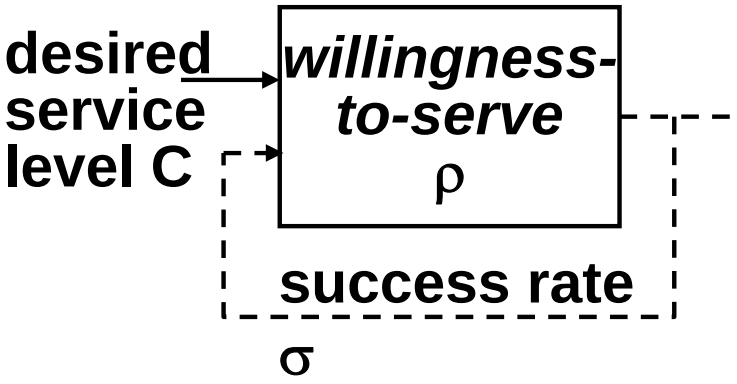
The matrix formulism here parallels to that used in ranking web pages (ref: Kleinberg HITS algorithm)

Reputation-based Admission

- Transactions → Credit Matrices → Reputations → 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

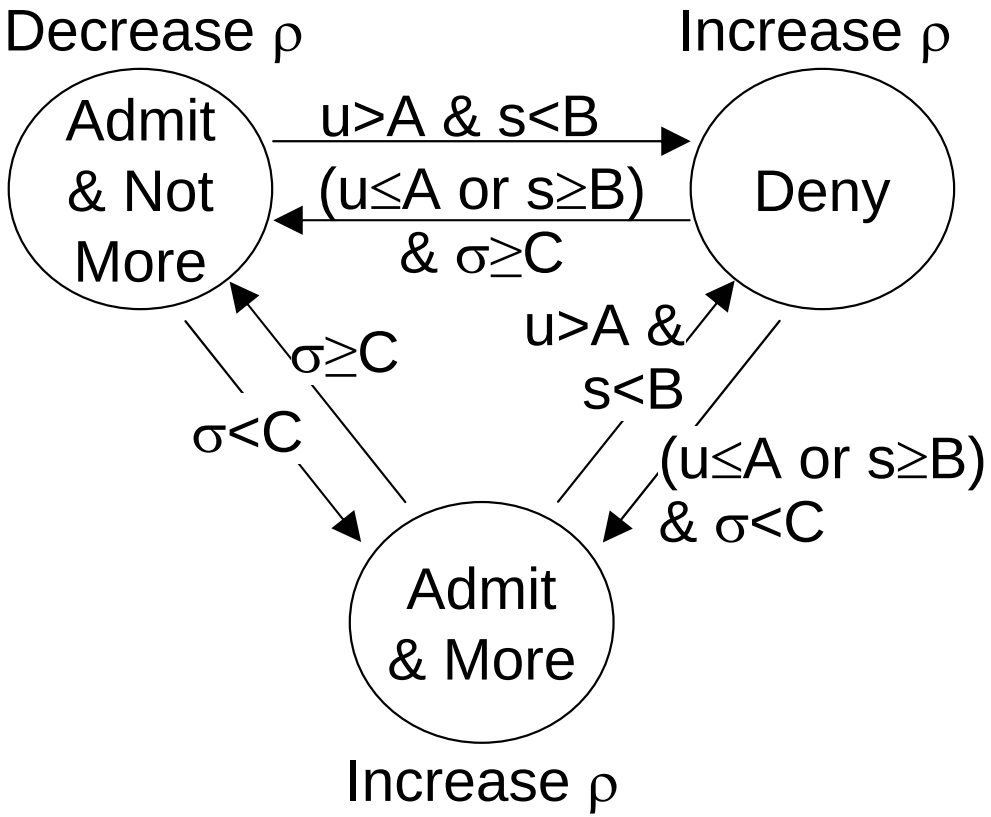
Goal: search for a minimum level of contribution a node needs to provide in order to receive a desired level of service from other nodes



- ***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-to-serve 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

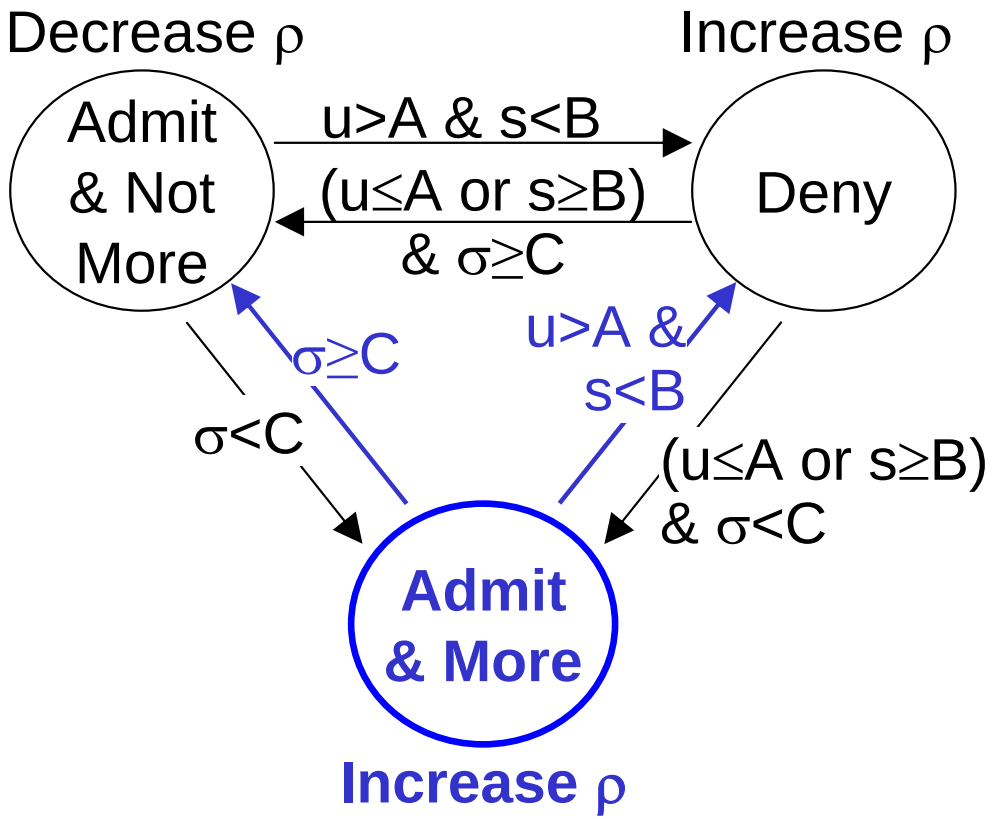
→ Some of these nodes may then enter the "**Deny**" state

→ 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.

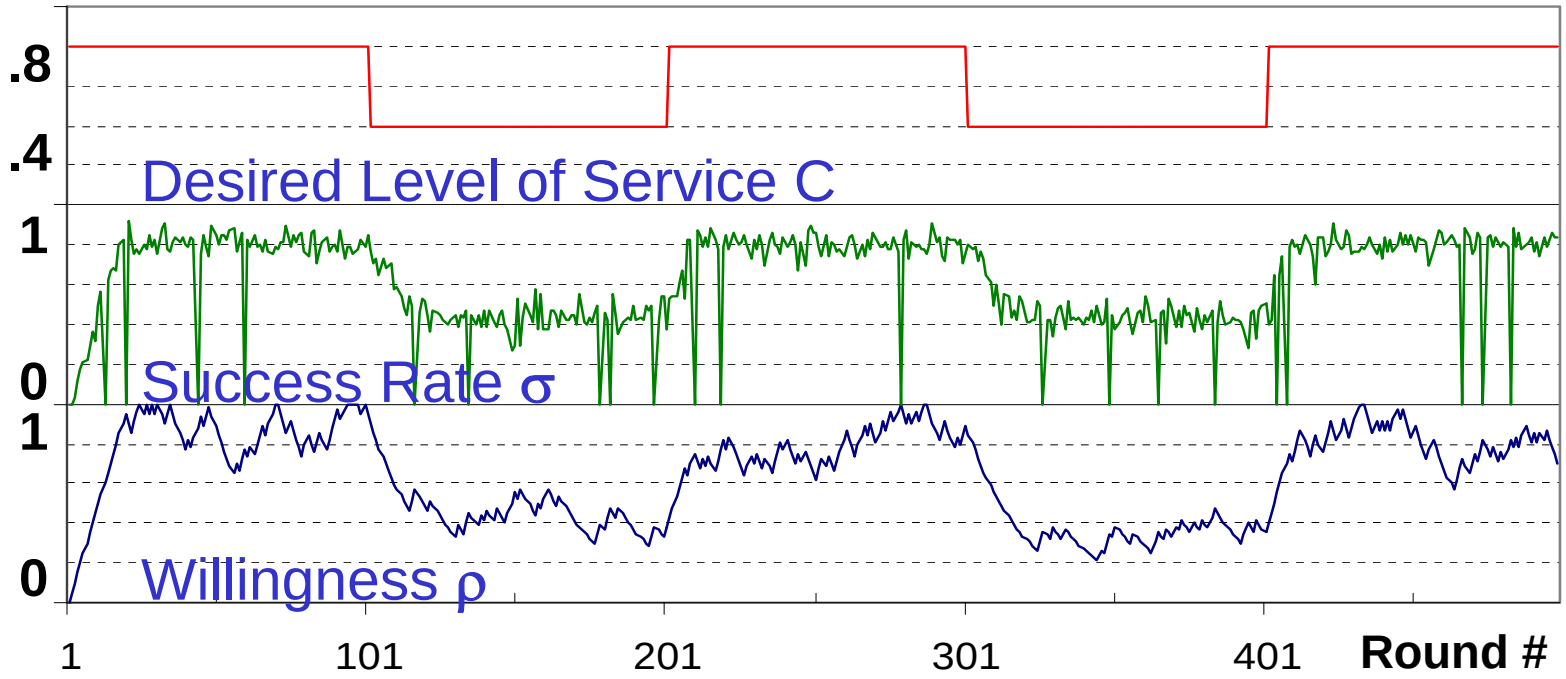


- If a new node makes many service requests such that its **u** is above **A**% and its **s** is below **B**%, it will enter the "**Deny**" state
- As long as its success rate σ does not reach its desired level of service **C**, it will stay at the "**Admit & More**" state and increase its ρ to raise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



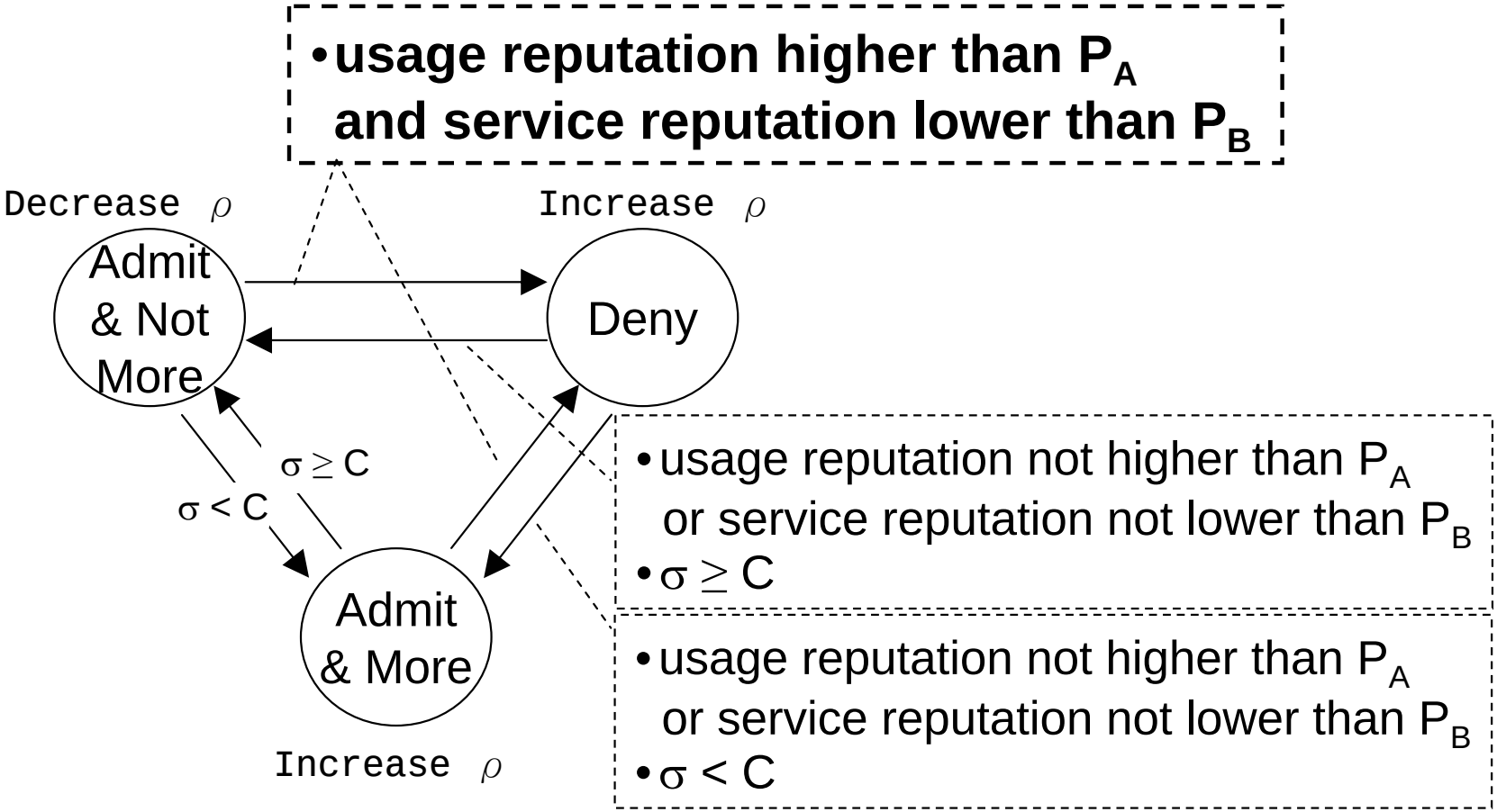
- **A** = .8, **B** = .2, **C** = .4 or .8, and # transactions per round = 10,000
- When in the "Admit & Not More" state, a node decreases ρ using ρ ← .95 * ρ, else it increases ρ using ρ ← max (ρ + .05, 1).Except when ρ = 1, the decreasing rate is smaller than the increasing rate
- The simulation results above show that
  - σ 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:

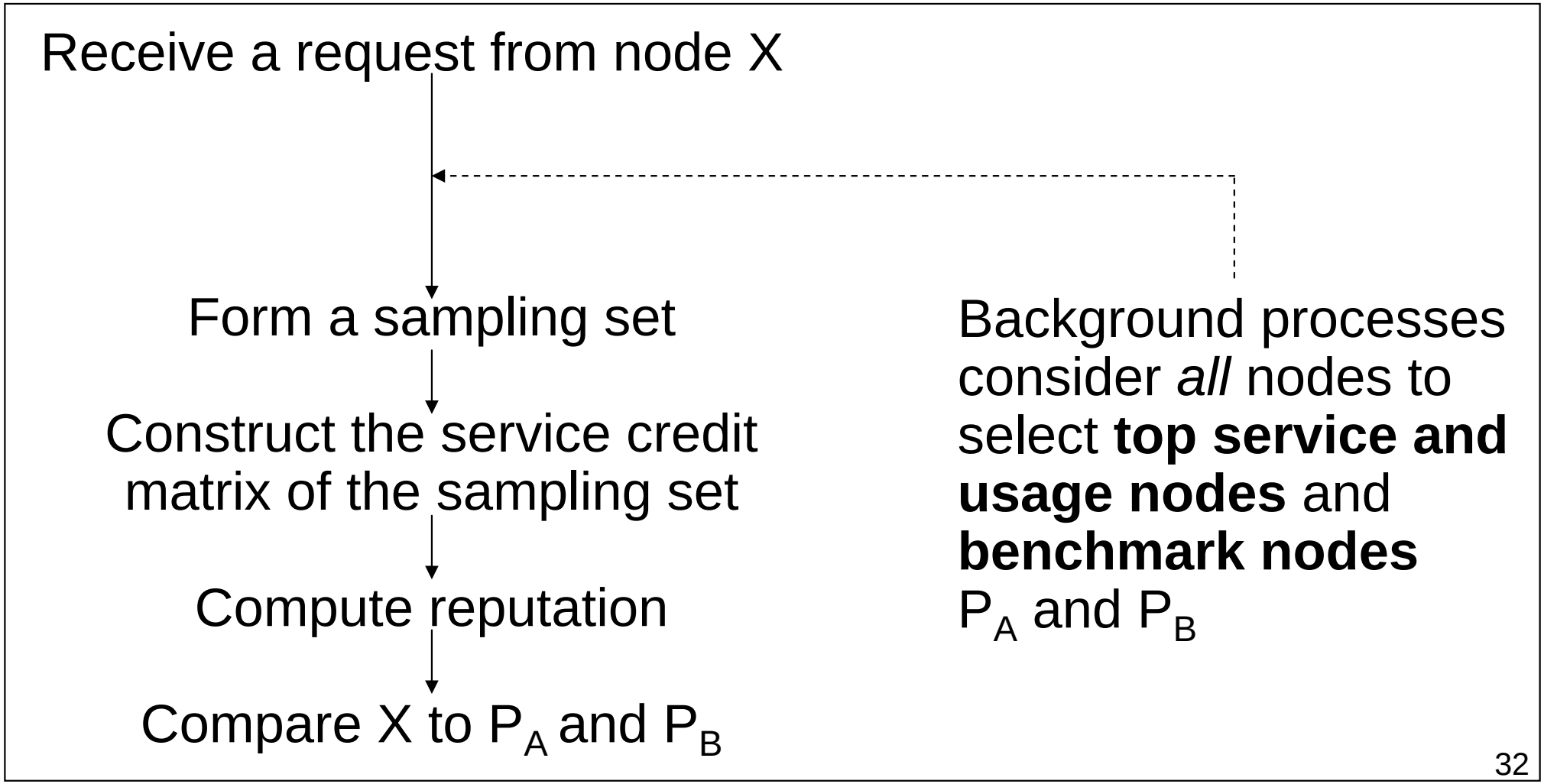
Node $\mathbf{P_A}$ has its usage-reputation ranking percentile at A% and
Node $\mathbf{P_B}$ has its service-reputation ranking percentile at B%

Need not know the exact rankings of a node.

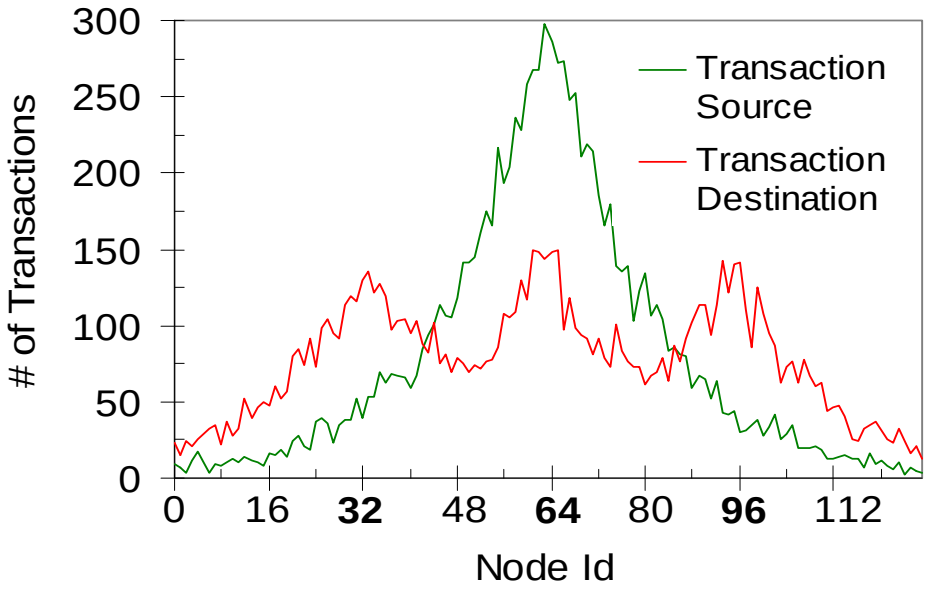


A Top-nodes Sampling Heuristic

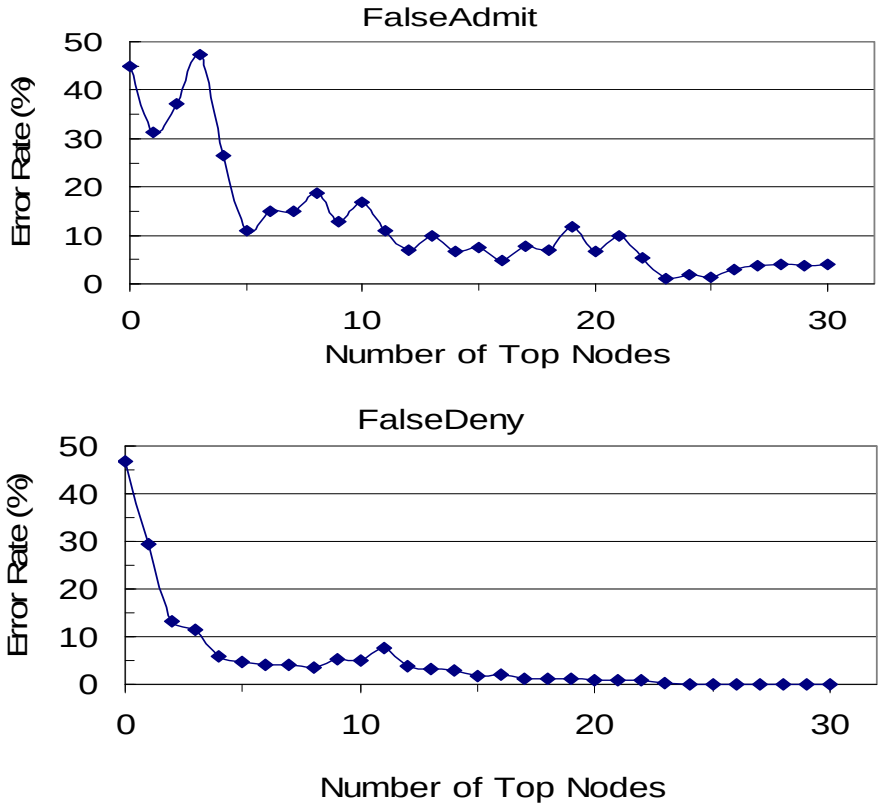
In computing reputation, we will only use a sampling set, consisting of top service and usage nodes, node X, and two benchmark nodes P_A and P_B



Effectiveness of the Sampling Heuristic: A Simulation Result

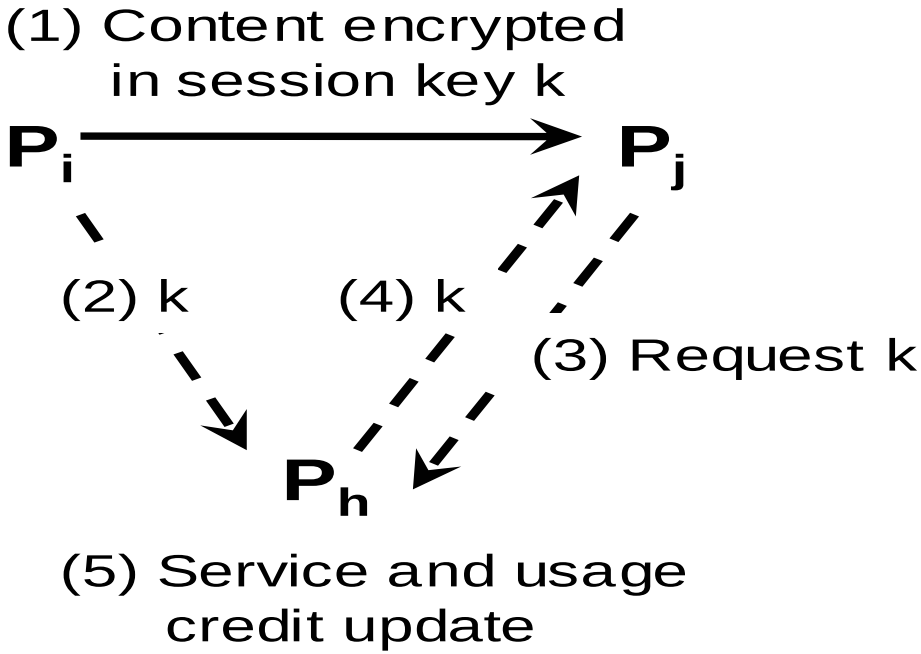


Load: mixtures of multiple Laplace distributions for requesting and requested nodes



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

Distributed Trust-enhancement Using a Third-party Node



- For a transaction of sending content from P_i to P_j
  - 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 of Differentiated Admission

- 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

Summary

- Self-organizing, autonomous peers of P2P systems introduce new applications, opportunities, and challenges
  - Fairness and trustworthy are two major concerns of P2P systems
- Incentives relieve the negative impacts of free-loaders
  - Payment-based approaches: PPay and KARMA
  - Reputation-based approaches: BitTorrent and Reputella
- Differentiated admission control mechanism incentivizes peers to control their resources
  - PeerRank identifies the reputations of peers