Erasure Codes for Heterogeneous Networked Storage Systems

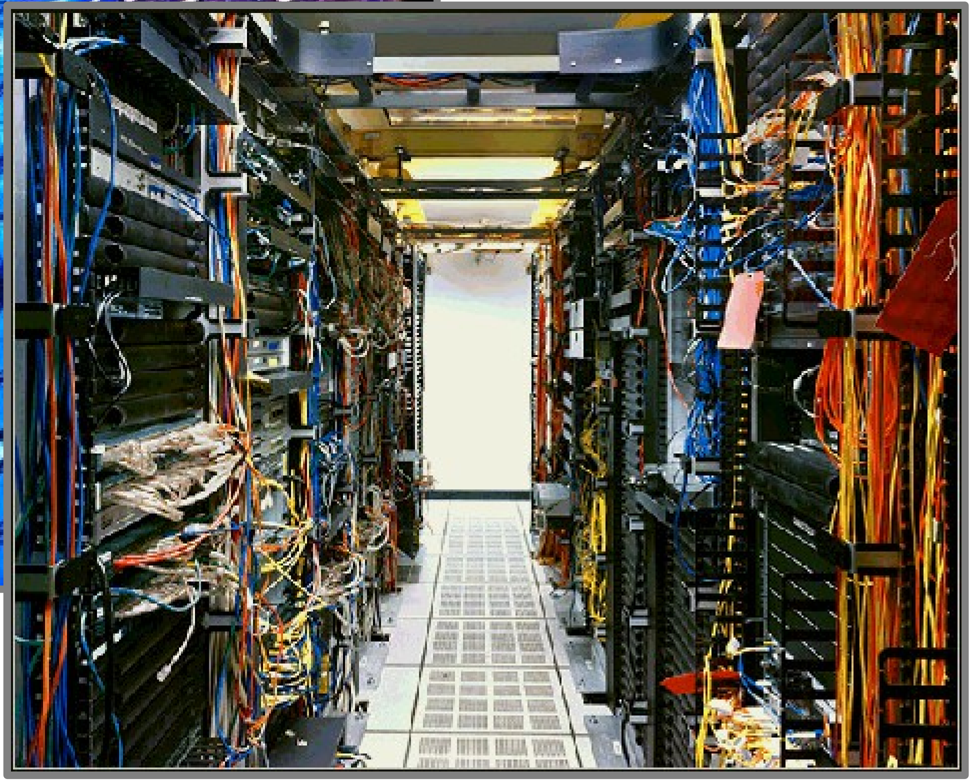
NANYANG TECHNOLOGICAL UNIVERSITY

Lluís Pàmies i Juárez
lpjuarez@ntu.edu.sg

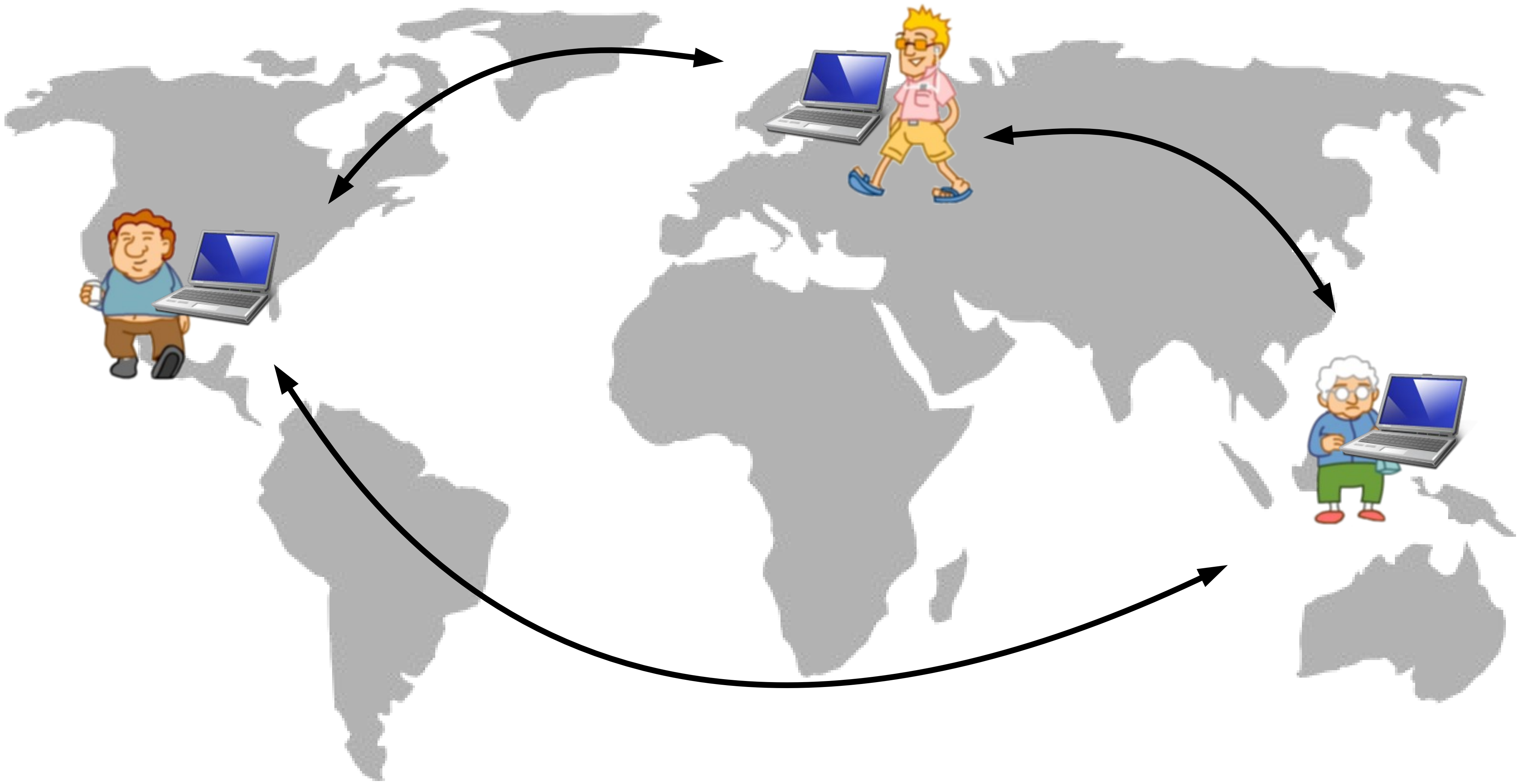
Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. Homogeneous Distributed Systems
4. Heterogeneous Distributed Systems:
    1. Orchestrated Systems
    2. P2P Systems
5. Other Open Problems in Distributed Storage Systems.

Data Centers



Peer-to-Peer / Friend-to-Friend Networks



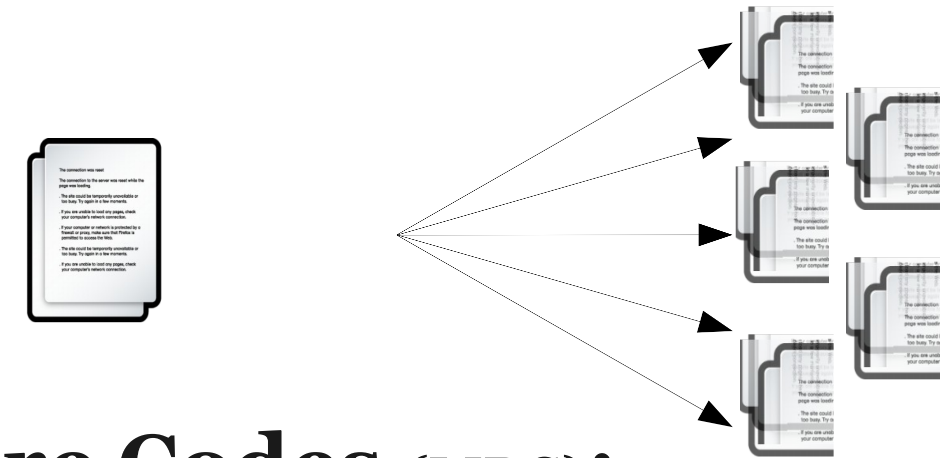
Networked Storage Systems

How to store a file maximizing its data reliability ?

Networked Storage Systems

Replication:

Networked Storage Systems

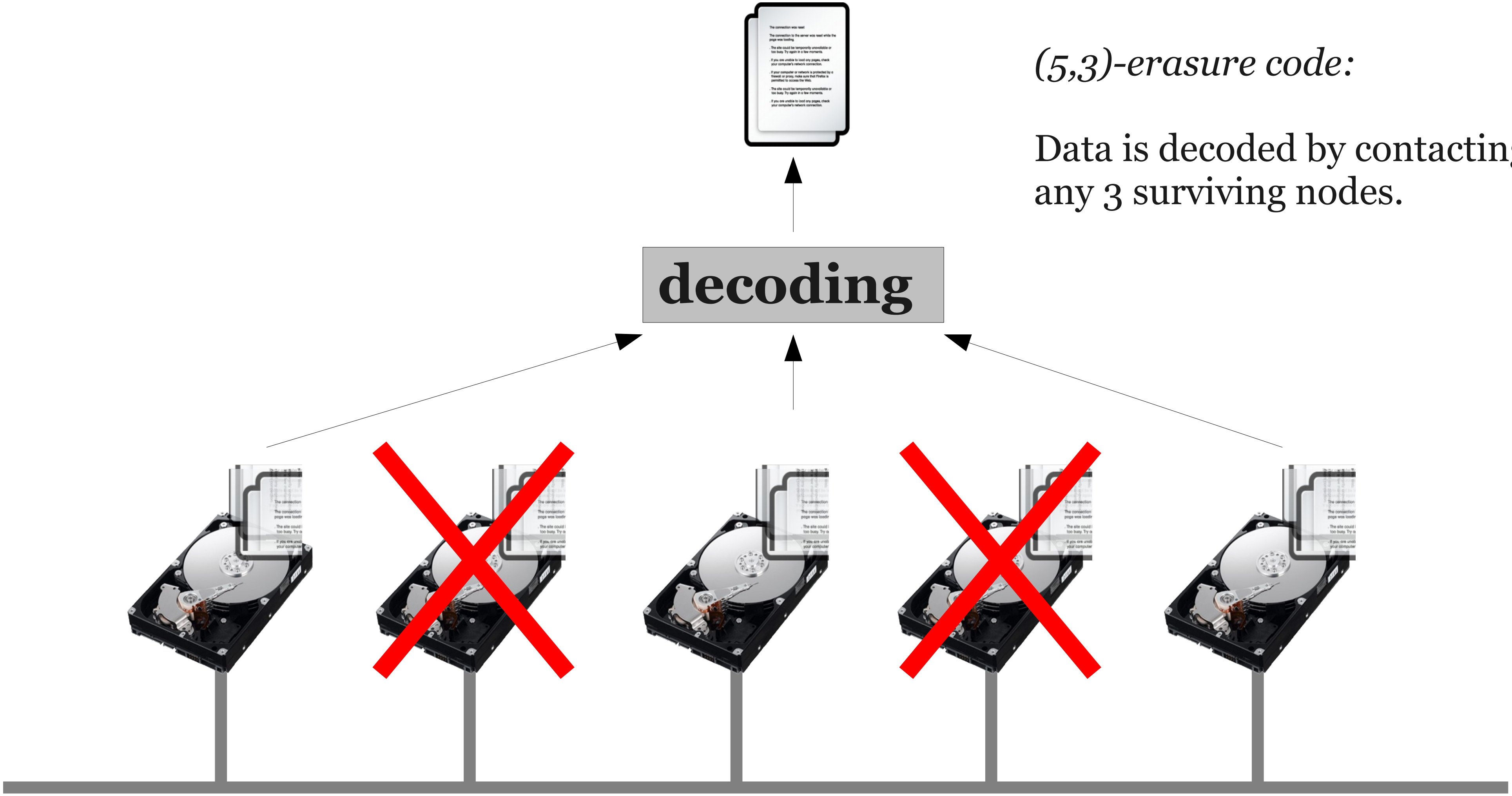


(5,3)-erasure code:

Split the file in 3 chunks and generate 5 linear combinations of them (e.g. reed-solomon).

Erasure Codes (MDS):

Networked Storage Systems

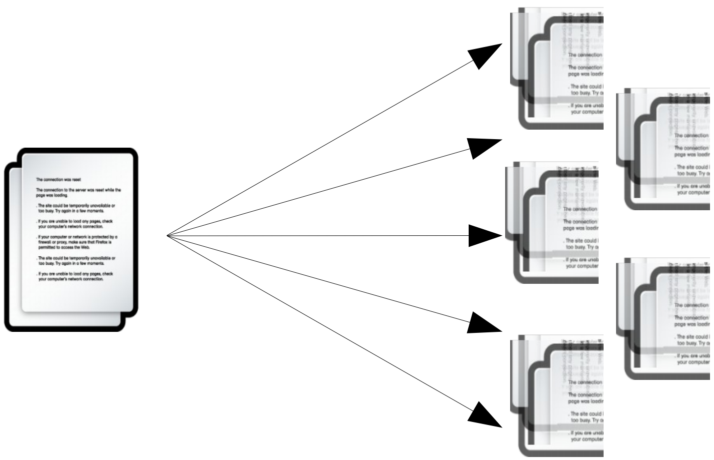


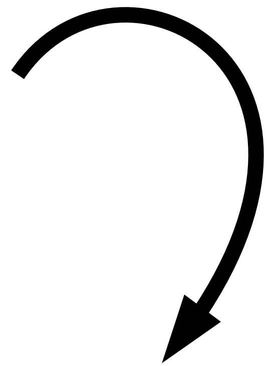
(5,3)-erasure code:

Data is decoded by contacting any 3 surviving nodes.

Same reliability with a smaller storage footprint (5/3 instead of 3)

Storage Allocation Problem





How to assign *n* redundant blocks to a given set of storage nodes ?

De-facto Premises

- Node failures/unavailabilities follow an uniform distribution.
- The assignment of the n encoded fragments has no impact on data reliability.

Traditional Coding

- Coding:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,k} \end{pmatrix} \cdot \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_k \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

- The encoded data stored to each node:
  - Has always the same size.
  - Has always the same importance.
  - Data assignment: 1 block per node

Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. **Homogeneous Distributed Systems**
4. Heterogeneous Distributed Systems:
    1. Orchestrated Systems
    2. P2P Systems
5. Other Open Problems in Distributed Storage Systems.

Exploiting Heterogeneities: Alternative Coding Schemes

- Different size:

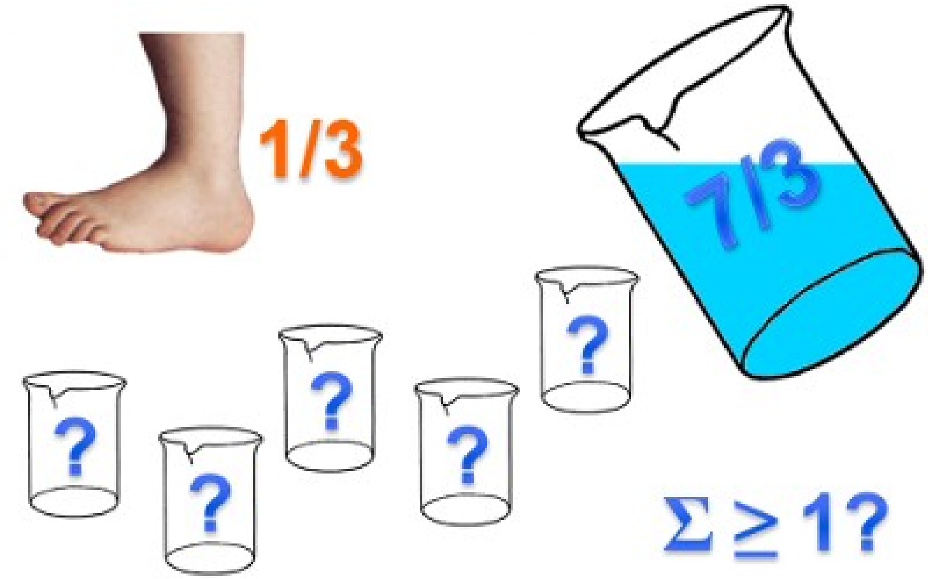
$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,k} \\ \vdots & \ddots & \ddots & \vdots \\ a_{2n,1} & a_{2n,2} & \cdots & a_{2n,k} \end{pmatrix}$$

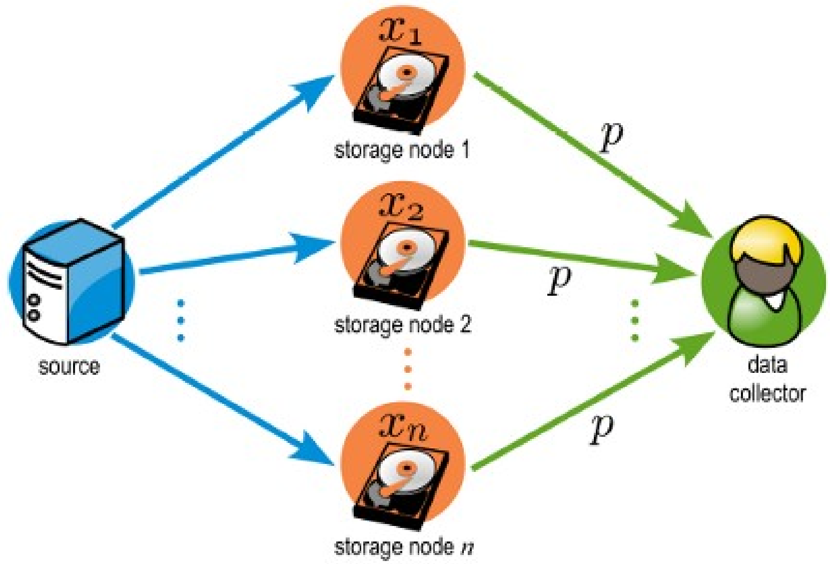
- Different importance[1]:

$$\begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & 0 & \ddots & \vdots \\ a_{n,1} & 0 & \cdots & a_{n,k} \end{pmatrix}$$

[1] Hierarchical Codes. *Duminuco, Biersack* (P2P'2008)

Storage Allocation Problem





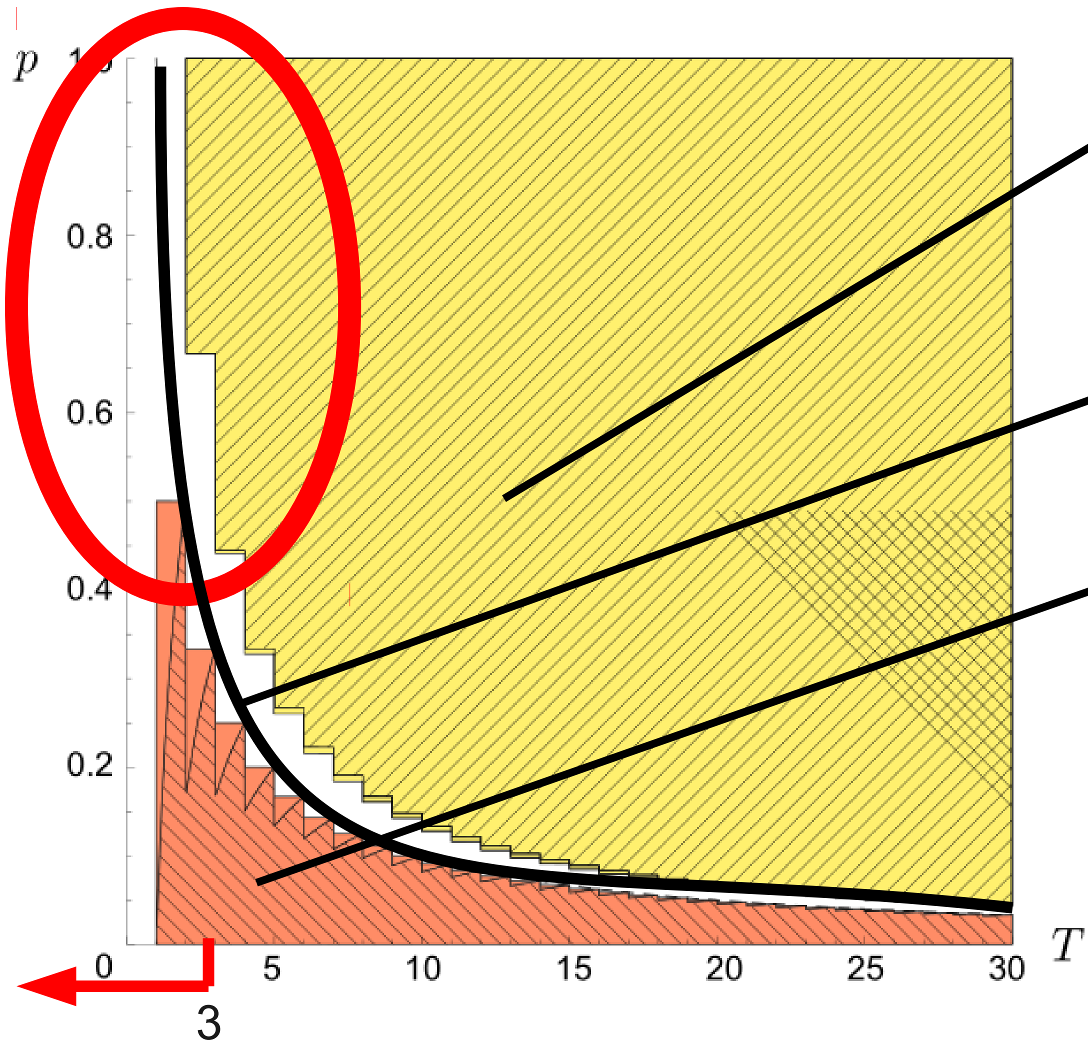
$$\left\{\frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\} \longrightarrow 0.90535$$

$$\left\{\frac{7}{6}, \frac{7}{6}, 0, 0, 0\right\} \longrightarrow 0.88889$$

$$\left\{\frac{7}{12}, \frac{7}{12}, \frac{7}{12}, \frac{7}{12}, 0\right\} \longrightarrow 0.88889$$

$$\left\{\frac{7}{15}, \frac{7}{15}, \frac{7}{15}, \frac{7}{15}, \frac{7}{15}\right\} \longrightarrow 0.79012$$

Solutions for Different Regimes (probabilistic access)



Max Symmetrical Spreading:
Spread the budget T uniformly into the n storage nodes.

$$p = \frac{1}{T}$$

Min Symmetrical Spreading:
Spread the budget T uniformly into T out of the n storage nodes.
REPLICATION

[1] Erasure code replication revisited . --- *Lin, Chiu, Lee*. P2P'2004
[2] Distributed Storage Allocation Problems. --- *Leong, Dimakis, Ho*. NetCod'2010
[3] Distributed Storage Allocation for High Reliability. --- *Leong, Dimakis, Ho*. ICC'2010
[4] Symmetric Allocations for Distributed Storage. --- *Leong, Dimakis, Ho*. GLOBECOM'2010

Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. Homogeneous Distributed Systems
4. **Heterogeneous Distributed Systems:**
    1. Orchestrated Systems
    2. P2P Systems
5. Other Open Problems in Distributed Storage Systems.

Problem in Heterogeneous Environments

- How are redundant blocks assigned to storage nodes?
  - uniformly?

    N=4
  - randomly?

    N=12
  - proportionally?

    N=12

Assignments in Different Scenarios

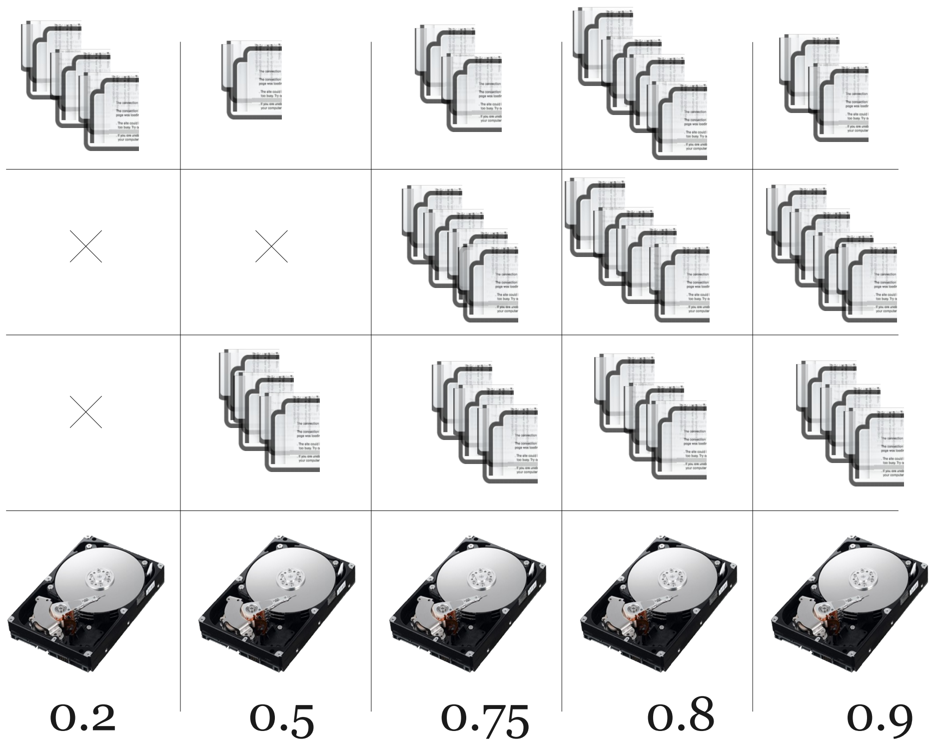
- Orchestrated Storage Systems:
  - All storage nodes belong to the same organization.
  - The objective is to maximize overall storage capacity.
- P2P Storage Systems:
  - Each node is a user that exchanges data reciprocally with other users → Users need to provide more resources to obtain more capacity.
  - Users aim to minimize the resources they have to exchange to store a given amount of data.

Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. Homogeneous Distributed Systems
4. Heterogeneous Distributed Systems:
    1. **Orchestrated Systems**
    2. P2P Systems
5. Other Open Problems in Distributed Storage Systems.

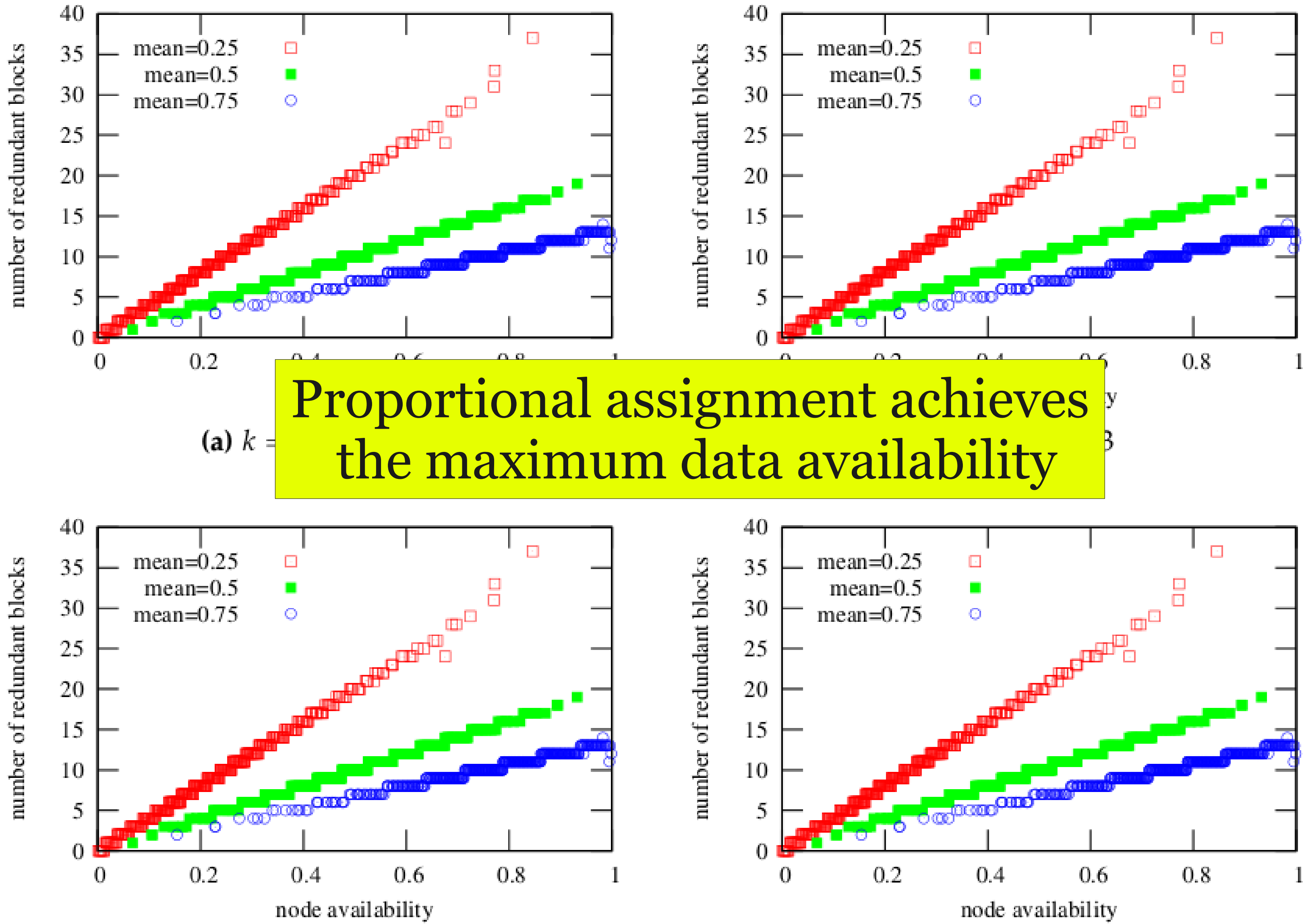
Assignment in Orchestrated Systems

- We generate a large number of redundant blocks and we try different assignment policies.
- Optimization process based on a PSO algorithm.
- We run the PSO for different redundancies, and different heterogeneities.
- Fitness function:
  - **data availability**



Number of nodes → 4
Number of blocks → 12

Assignment in Orchestrated Systems (cont'd)



Proportional assignment achieves the maximum data availability

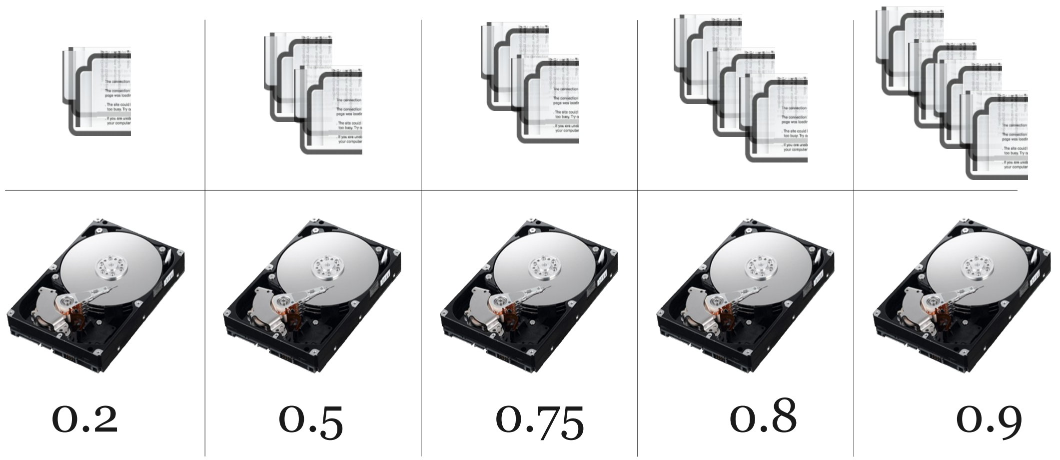
(c) $k = 500 = n/2$

(d) $k = 666 \approx n/1.5$

N=1000, nodes=100

Assignment in Orchestrated Systems (cont'd)

- Proportional assignment:



- Possible problems:
  1) Highest available nodes are over utilized
  2) Part of the capacity from lowest available nodes is never used → it minimizes the overall storage capacity.
  3) Unfair assignments in P2P

It does not happen!

Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. Homogeneous Distributed Systems
4. Heterogeneous Distributed Systems:
    1. Orchestrated Systems
    2. **P2P Systems**
5. Other Open Problems in Distributed Storage Systems.

Guaranteeing Fairness Between Peers

- Common decentralized solution to guarantee fairness among users:

  Reciprocal data exchanges between users.

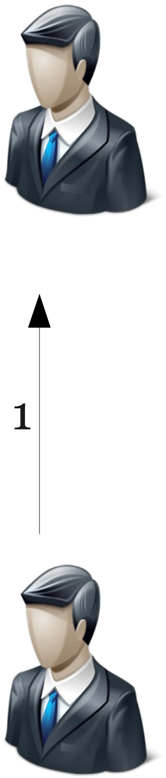
Guaranteeing Fairness Between Peers

- Common decentralized solution to guarantee fairness among users:

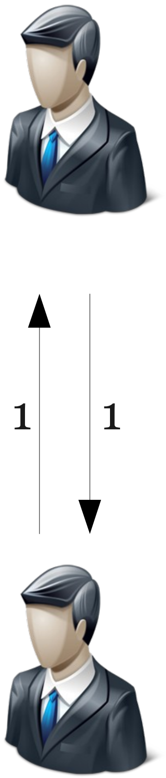
  Reciprocal data exchanges between users.

- Advantages:
  - **No third parties involved:**
    - Users need to find their own storage partners → they send data directly to the node that will store it.
  - **Fairness:**
    - For each data block stored remotely, peers needs to give back the same amount of local disk resources.
    - No users consumes more resources than the ones it provides.

Reciprocal Exchanges

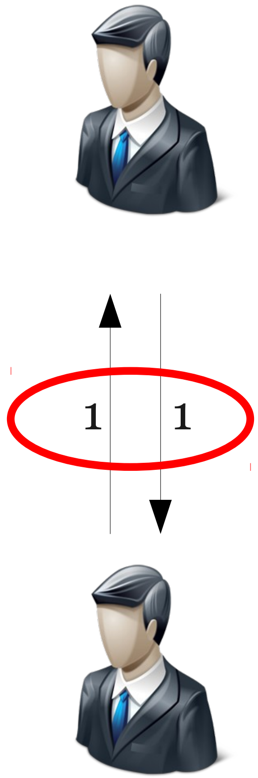


Reciprocal Exchanges

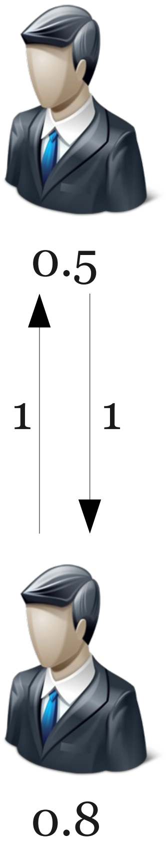


Reciprocal Exchanges

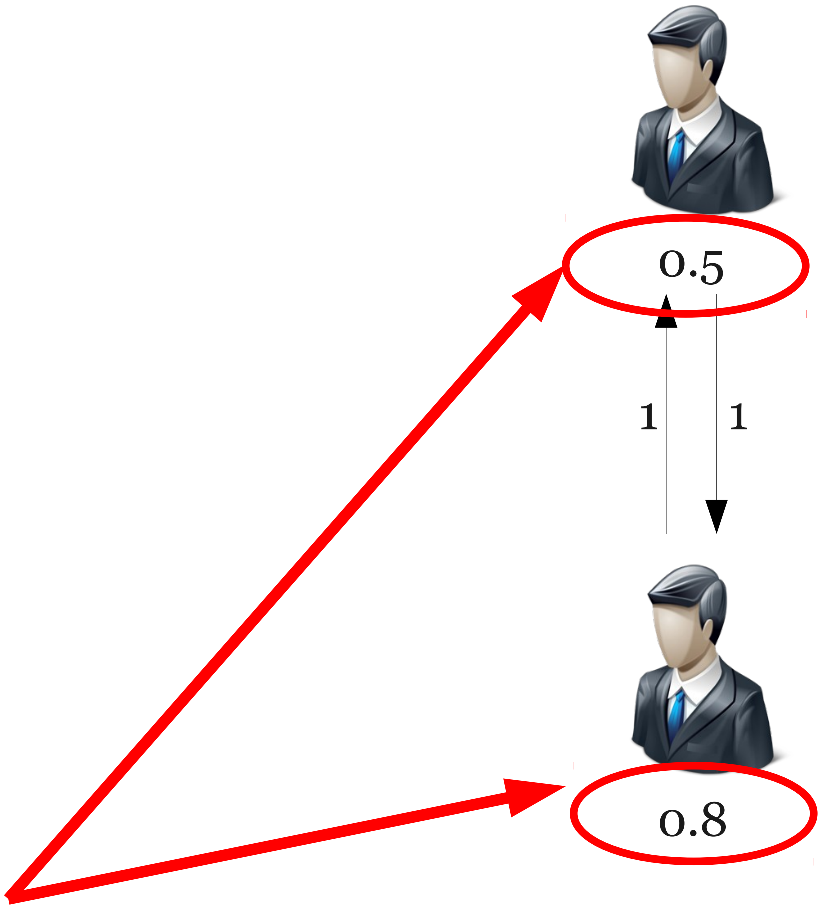
Symmetric
Exchanges



Reciprocal Exchanges

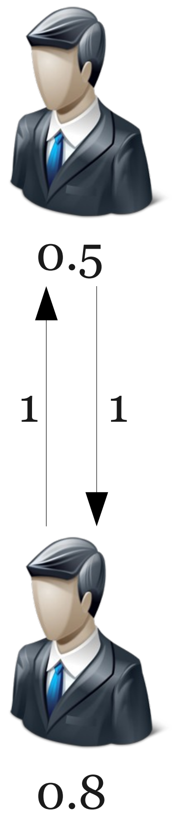


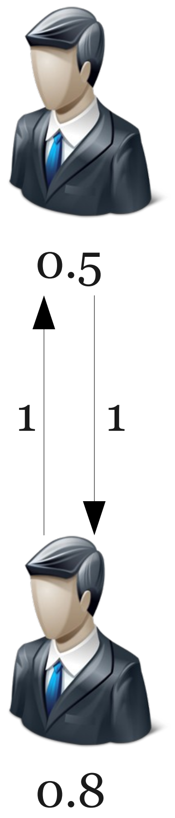
Reciprocal Exchanges



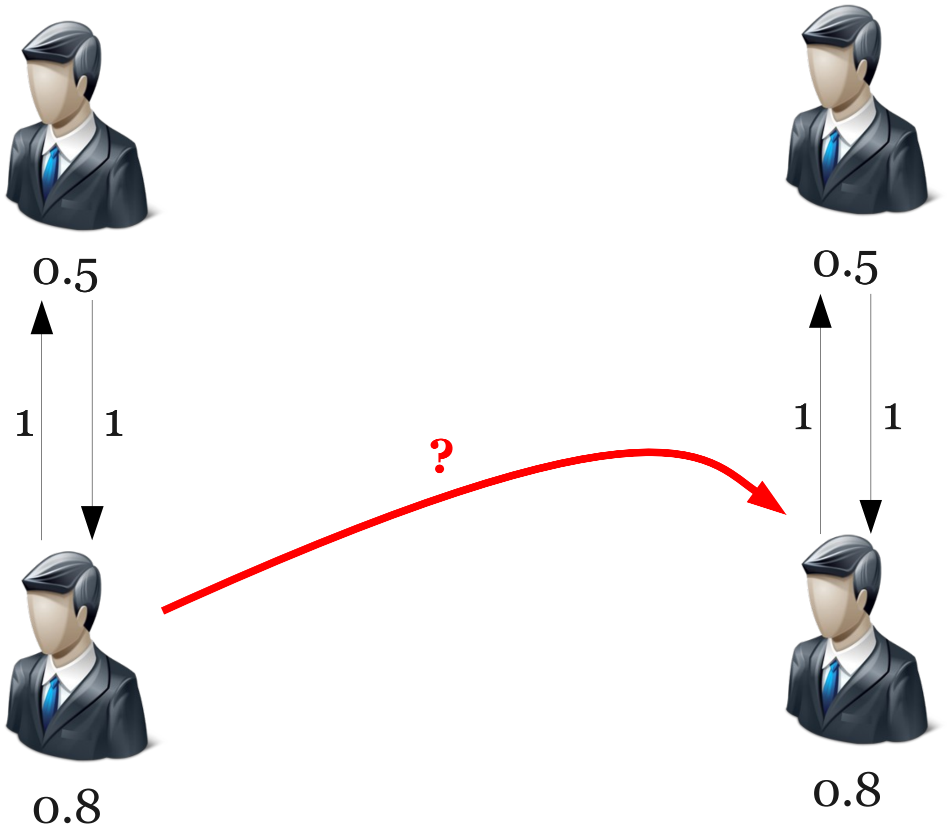
Problem: Does not incentivize users to improve their online availabilities.
Solution: Selfish partner selection.

Selfish Partner Selection

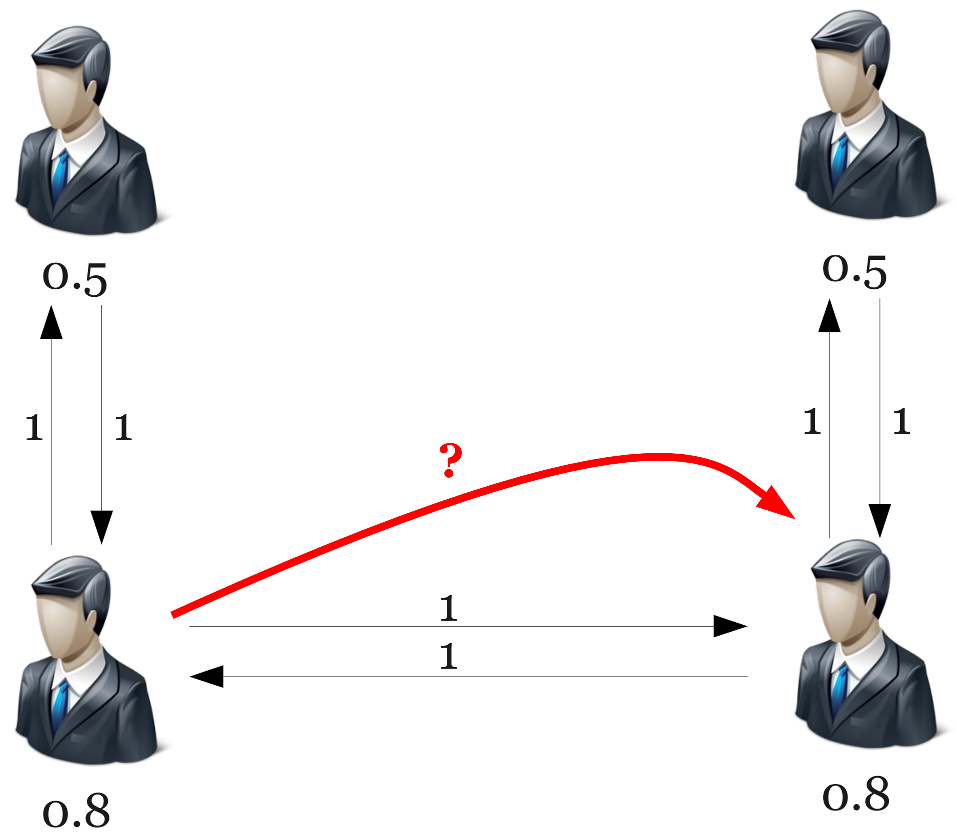




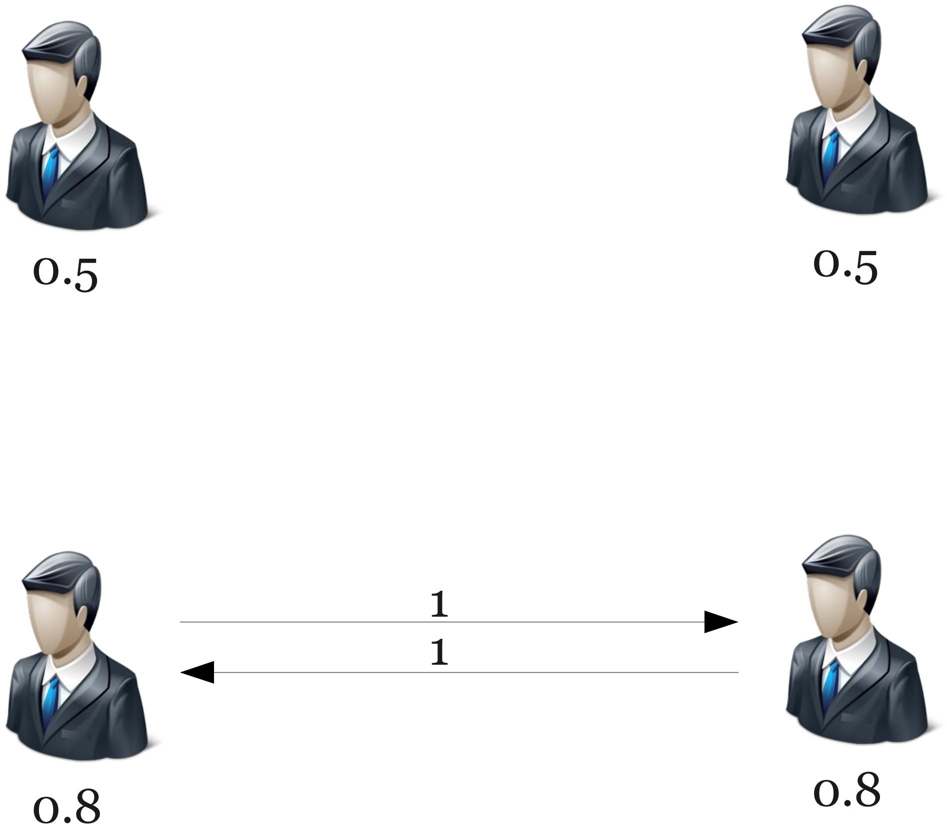
Selfish Partner Selection



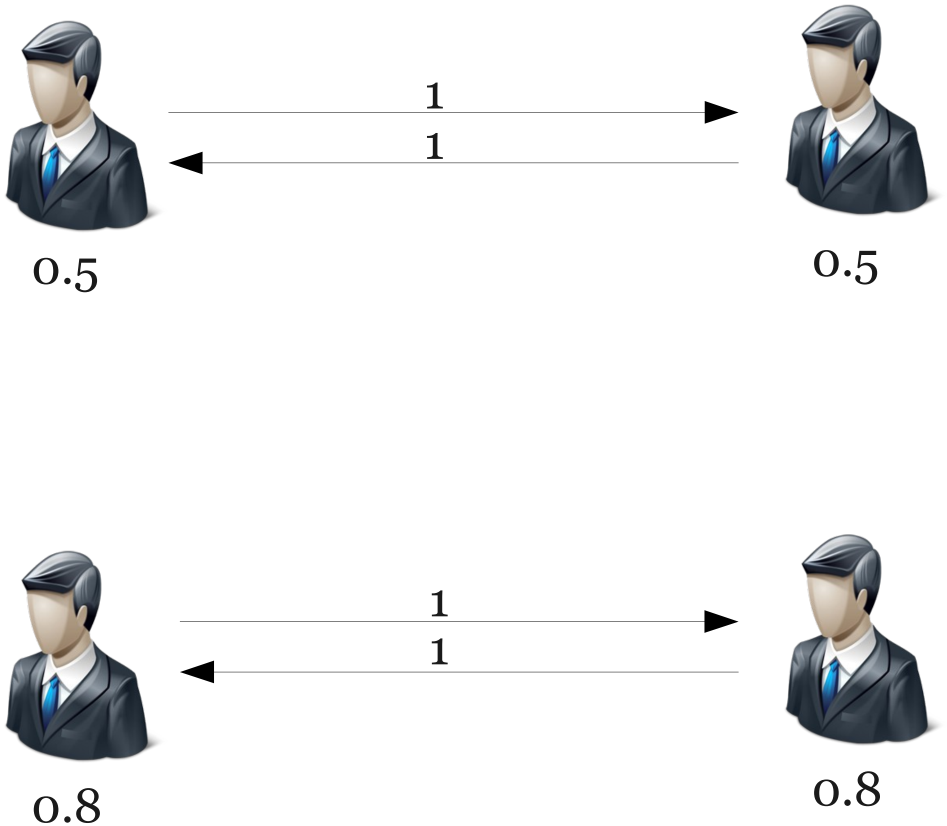
Selfish Partner Selection



Selfish Partner Selection



Selfish Partner Selection

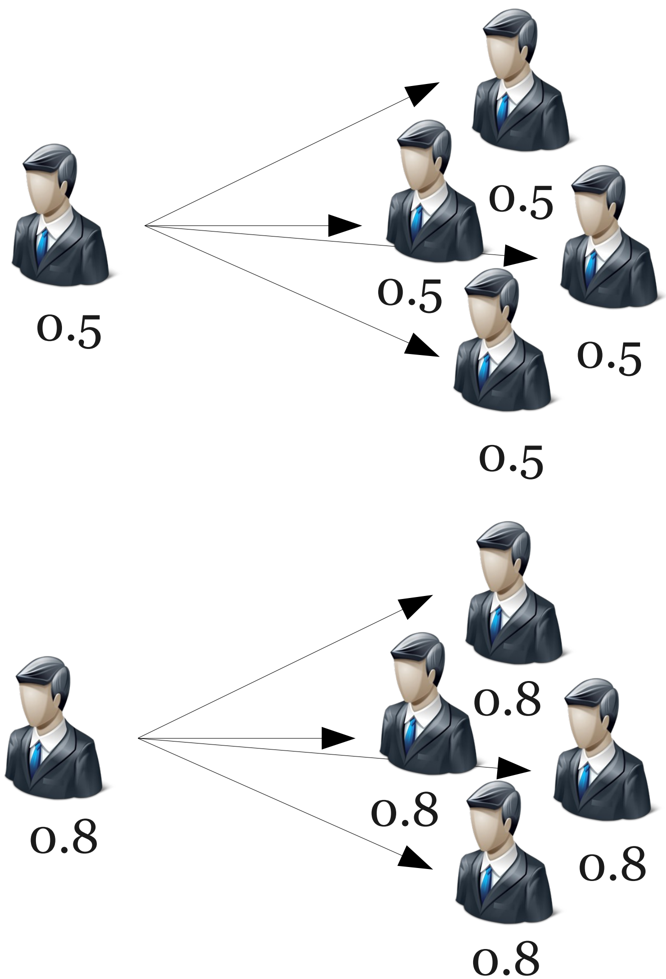


Gradient Topology: Users exchange data with users of similar online availability. High-available users require less redundancy.

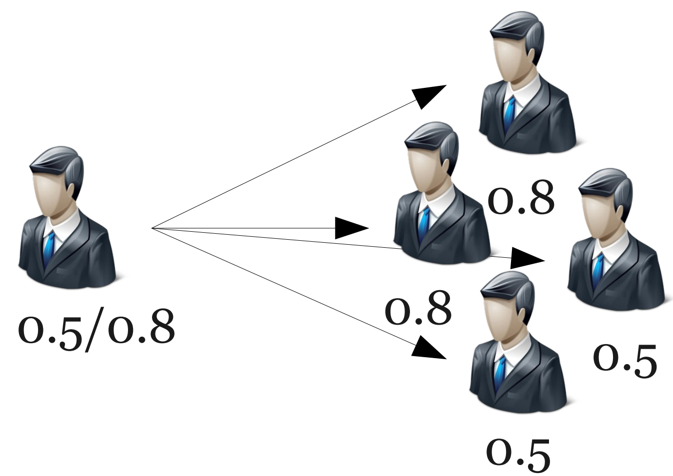
Problem With Selfish Partner Selection

- Lets compare two different scenarios:

Selfish Selection:



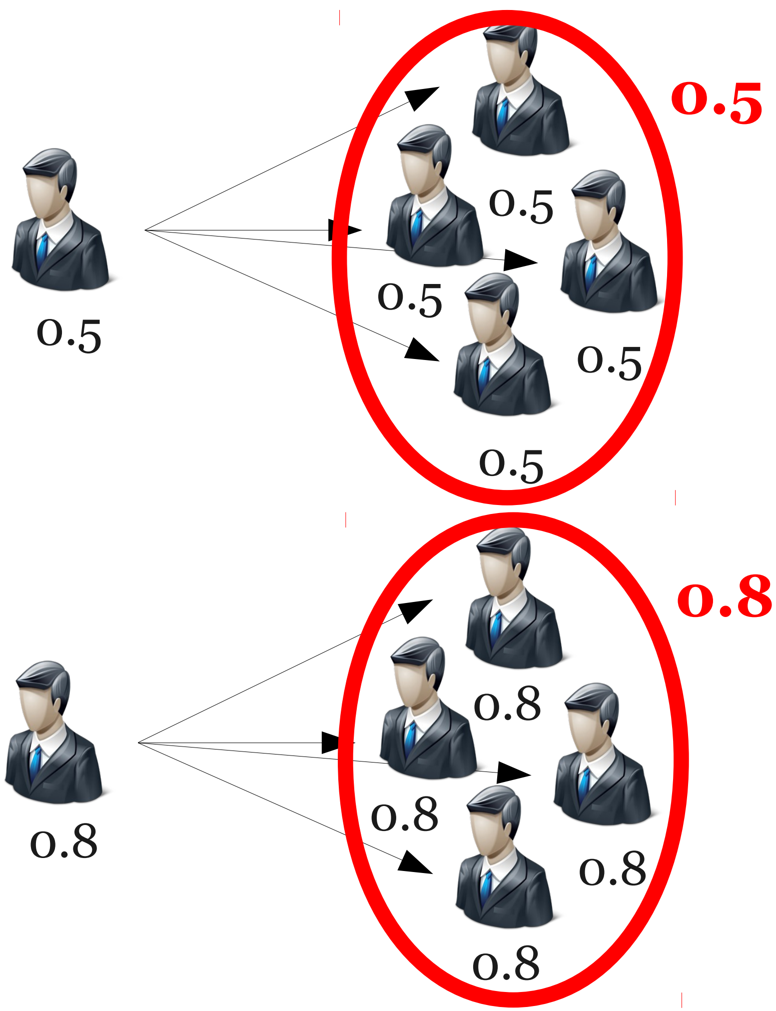
Random Selection:



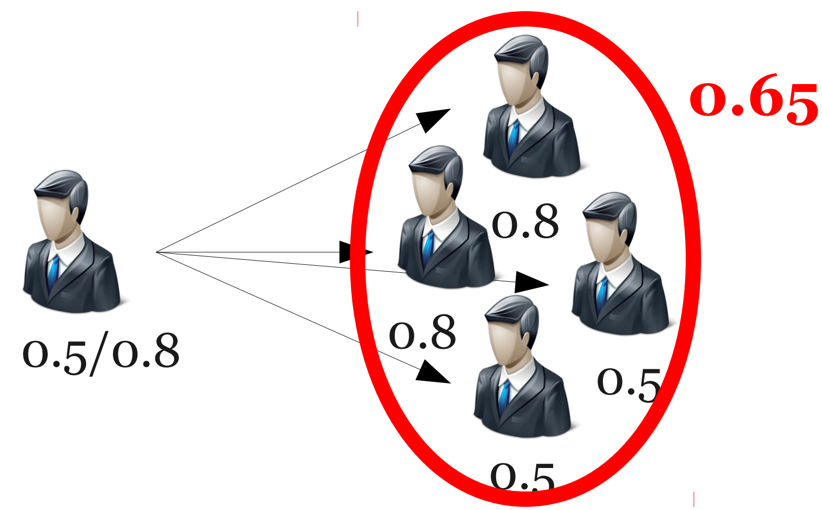
Problem With Selfish Partner Selection

- Lets compare two different scenarios:

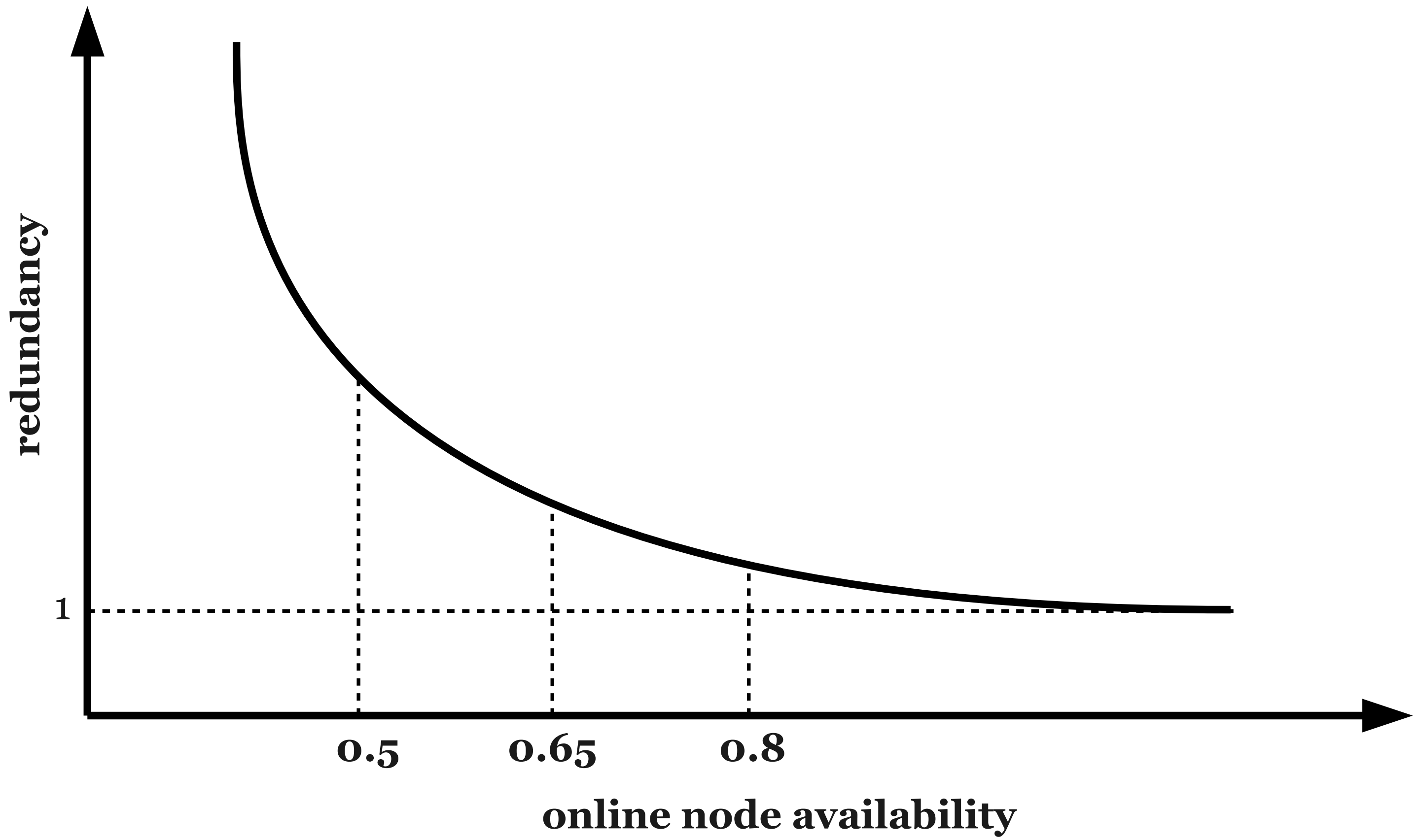
Selfish Selection:



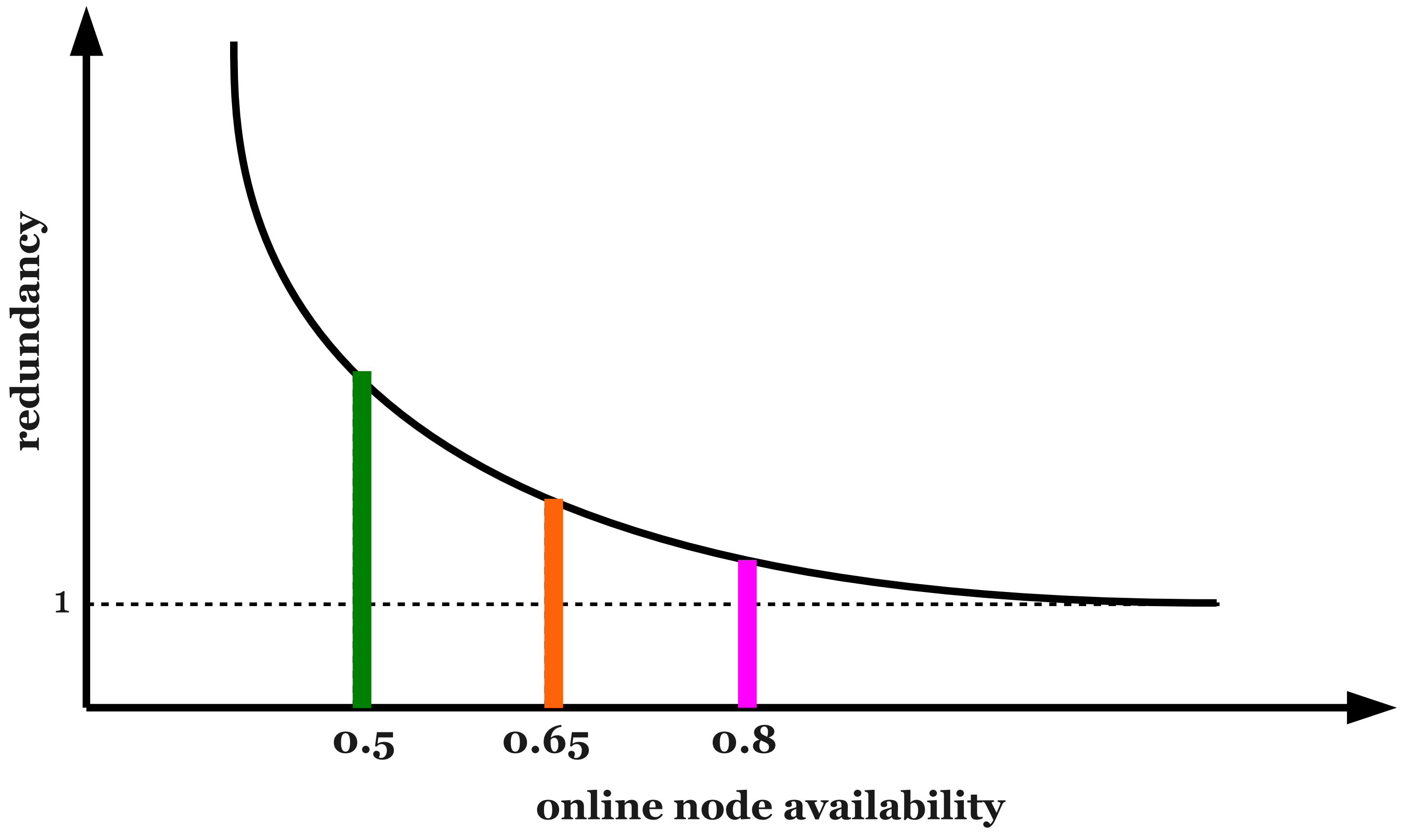
Random Selection:



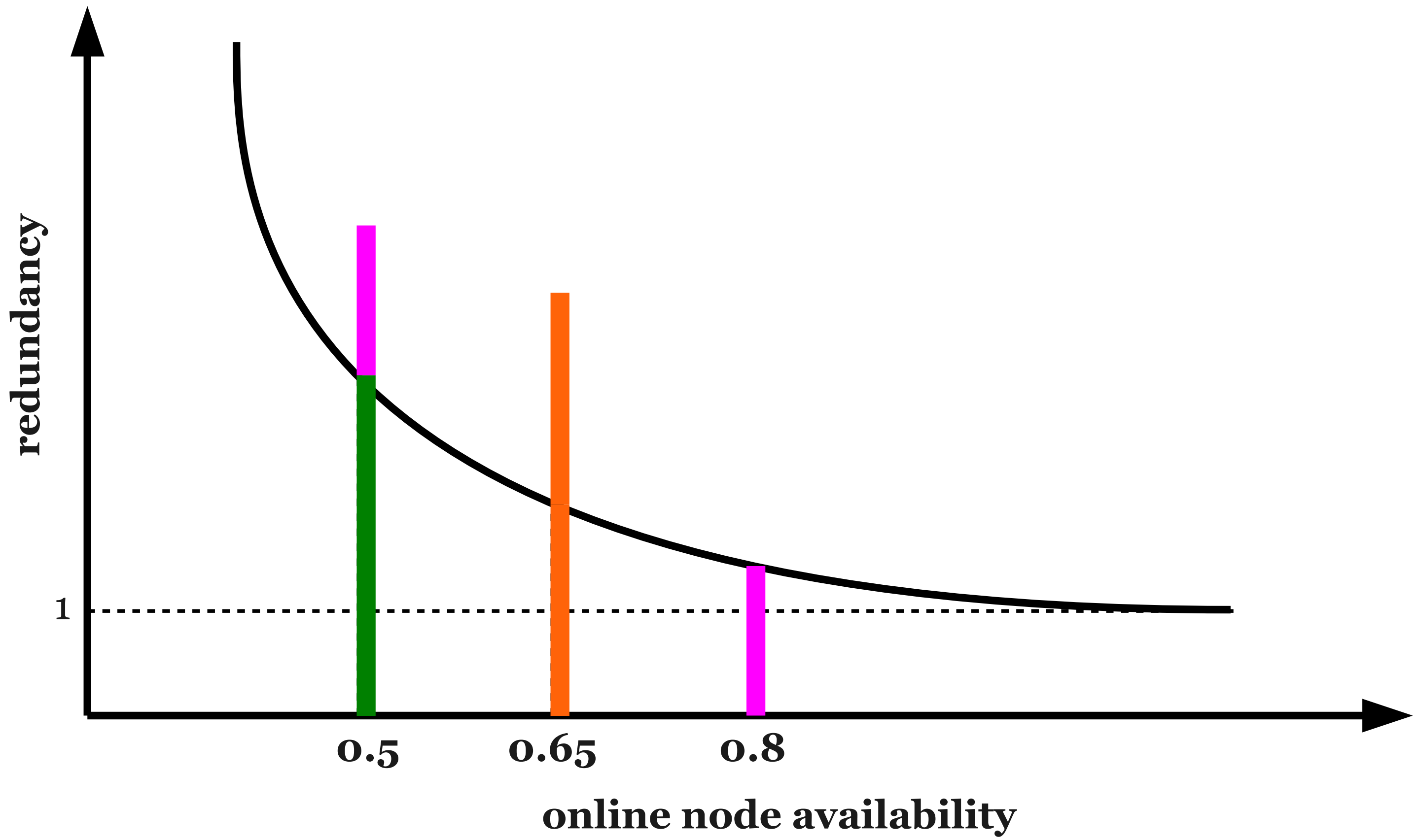
Problem With Selfish Partner Selection



Problem With Selfish Partner Selection

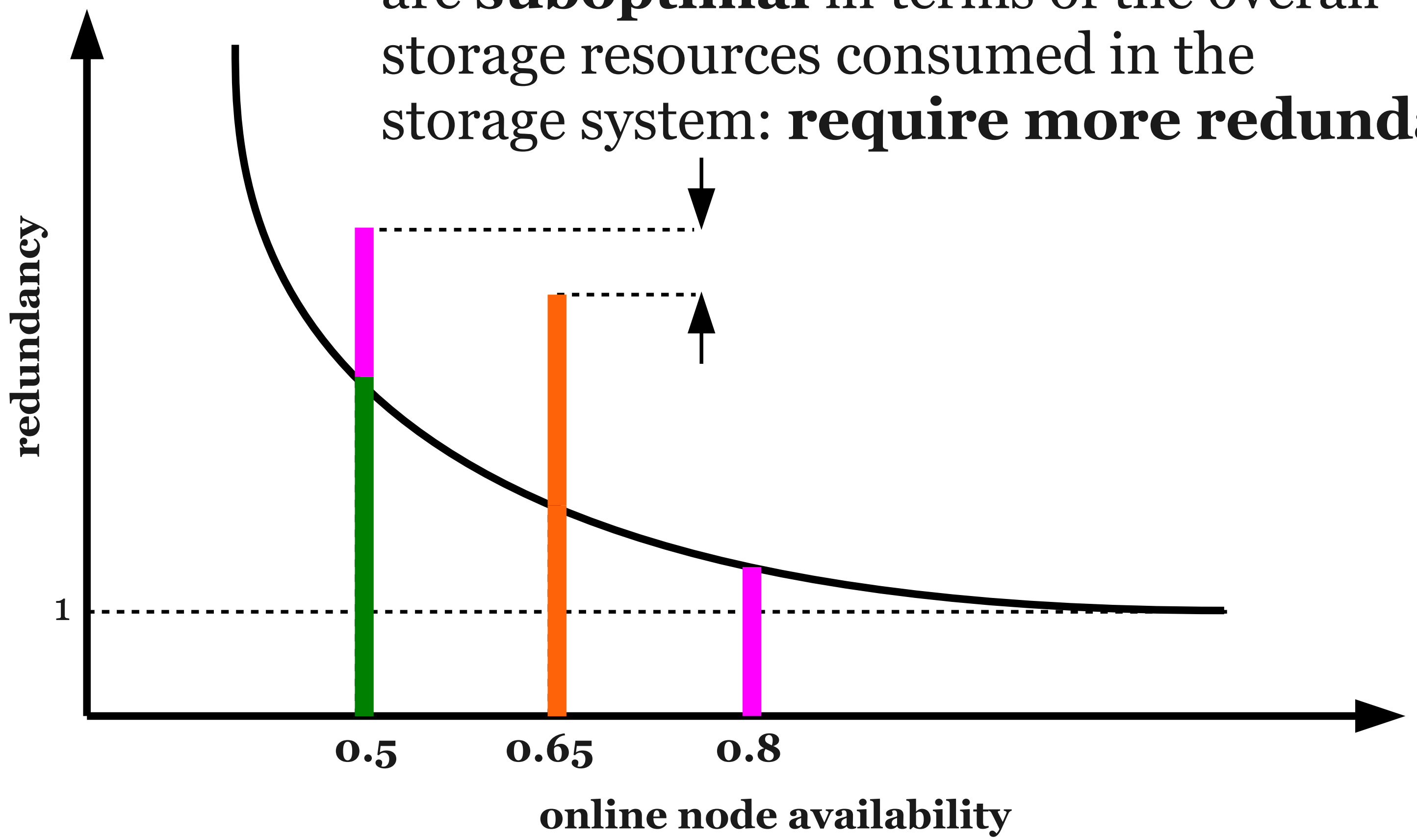


Problem With Selfish Partner Selection



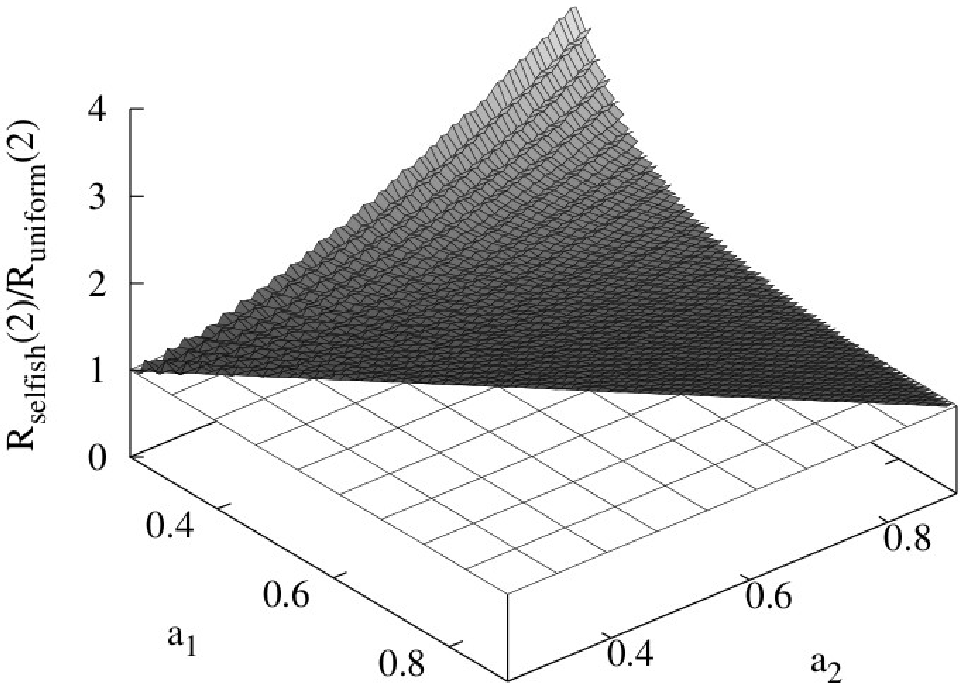
Problem With Selfish Partner Selection

Selfish selection and gradient topologies are **suboptimal** in terms of the overall storage resources consumed in the storage system: **require more redundancy.**



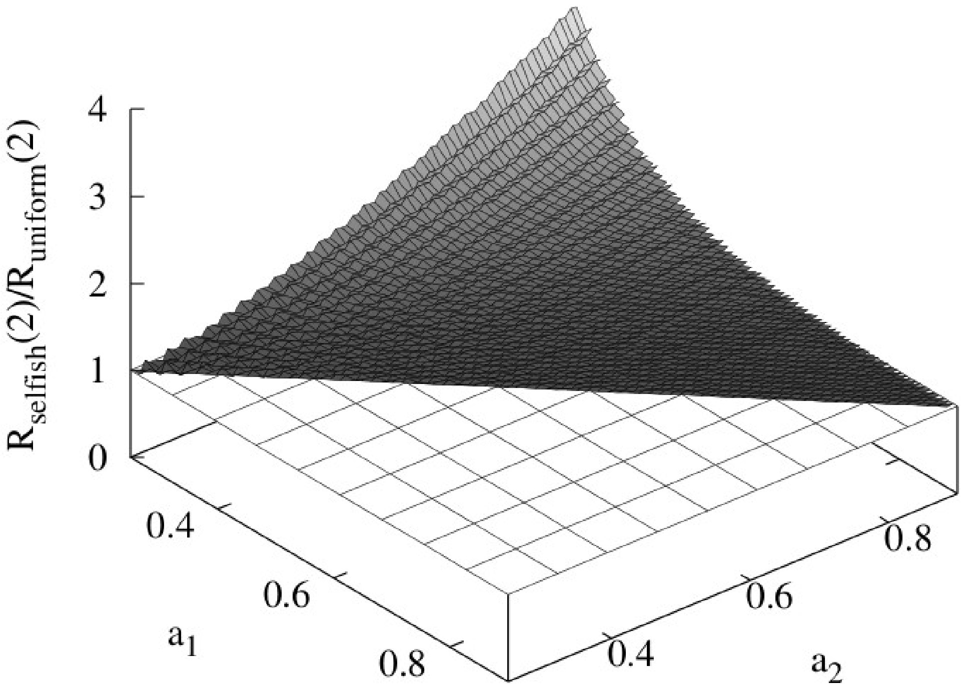
Problem With Selfish Partner Selection

Two nodes, all availabilities:

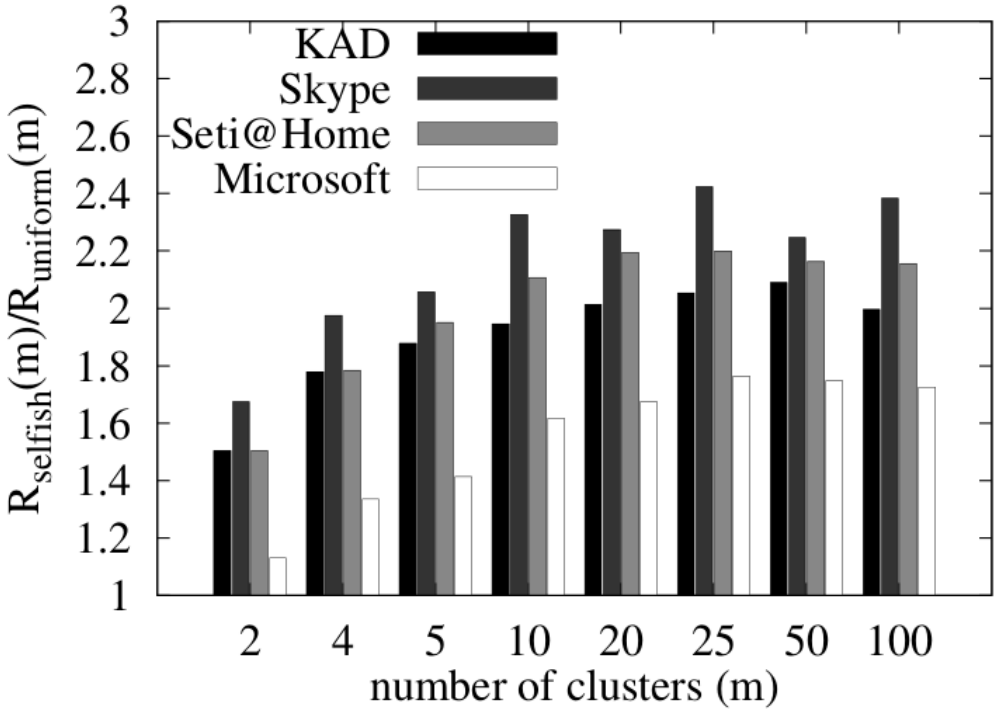


Problem With Selfish Partner Selection

Two nodes, all availabilities:



100 nodes, clustered by availability:



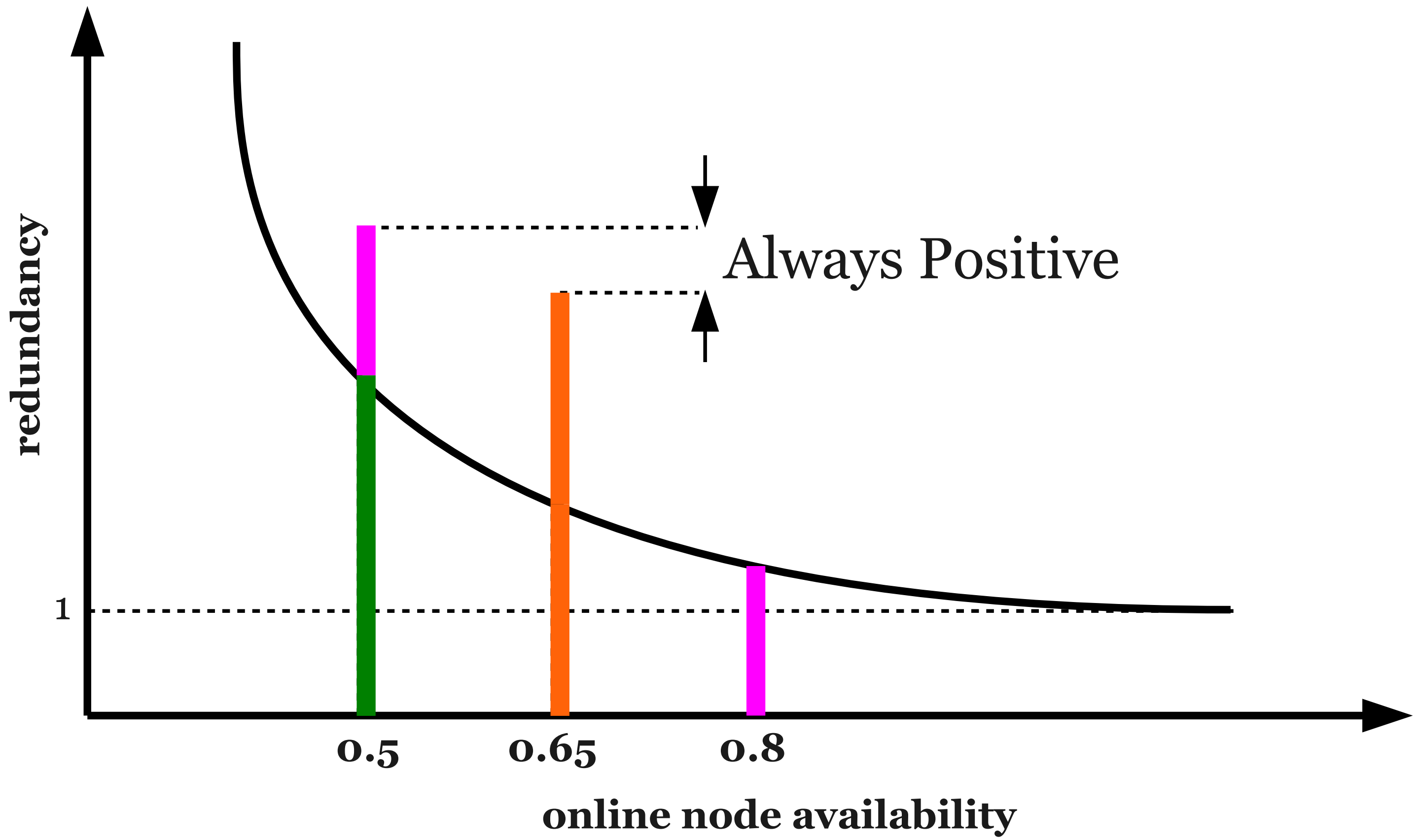
Problem Statement

- Random selection of storage partners reduces the overall storage resources required in the system:
  - Low available peers benefit by switching from selfish to random partner selection.
  - But high available peers are not interested on switching from selfish to random partner selection policy.

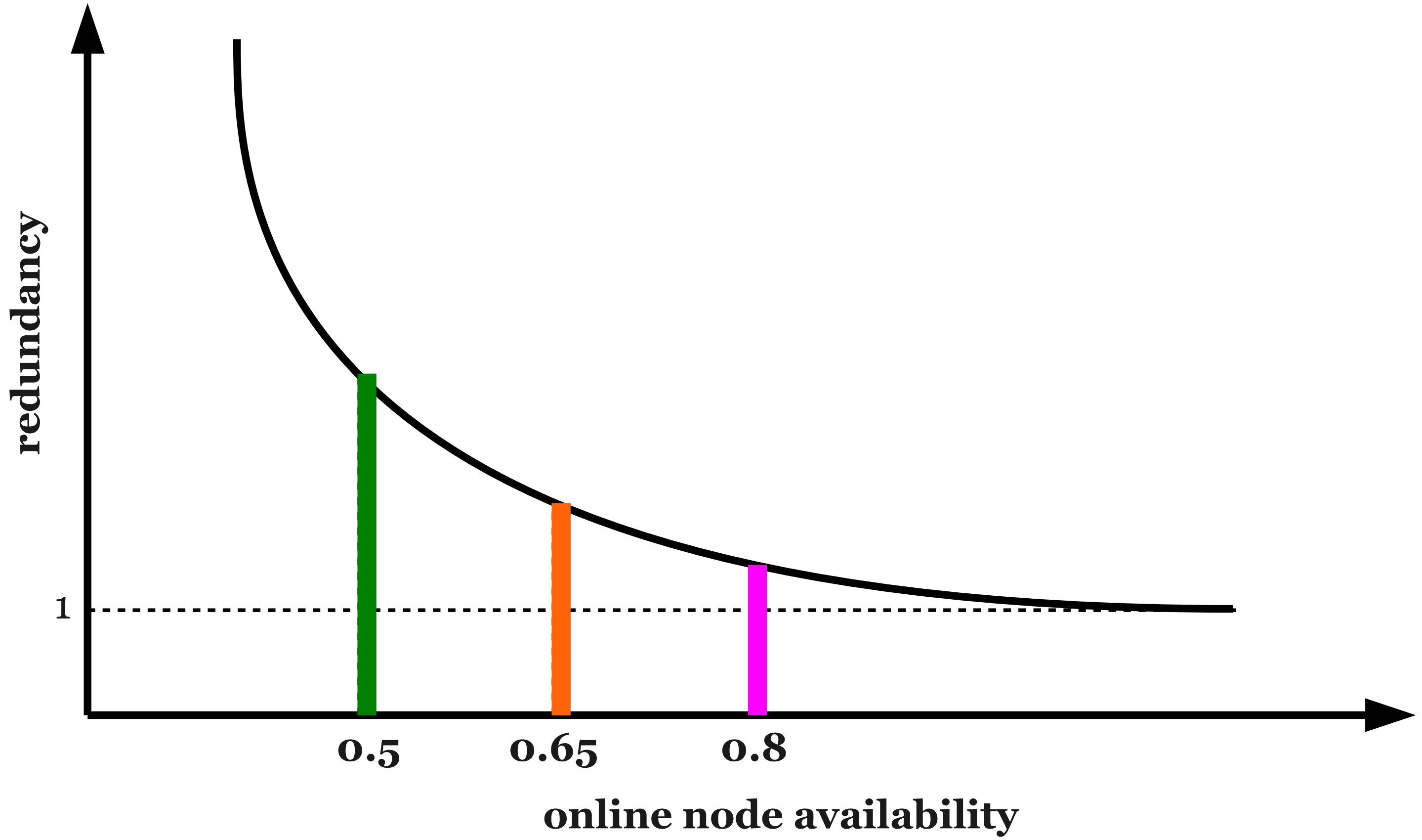
Problem Statement

- Random selection of storage partners reduces the overall storage resources required in the system:
  - Low available peers benefit by switching from selfish to random partner selection.
  - But high available peers are not interested on switching from selfish to random partner selection policy.
- **Can we make the random partner selection policy attractive for all peers ?**

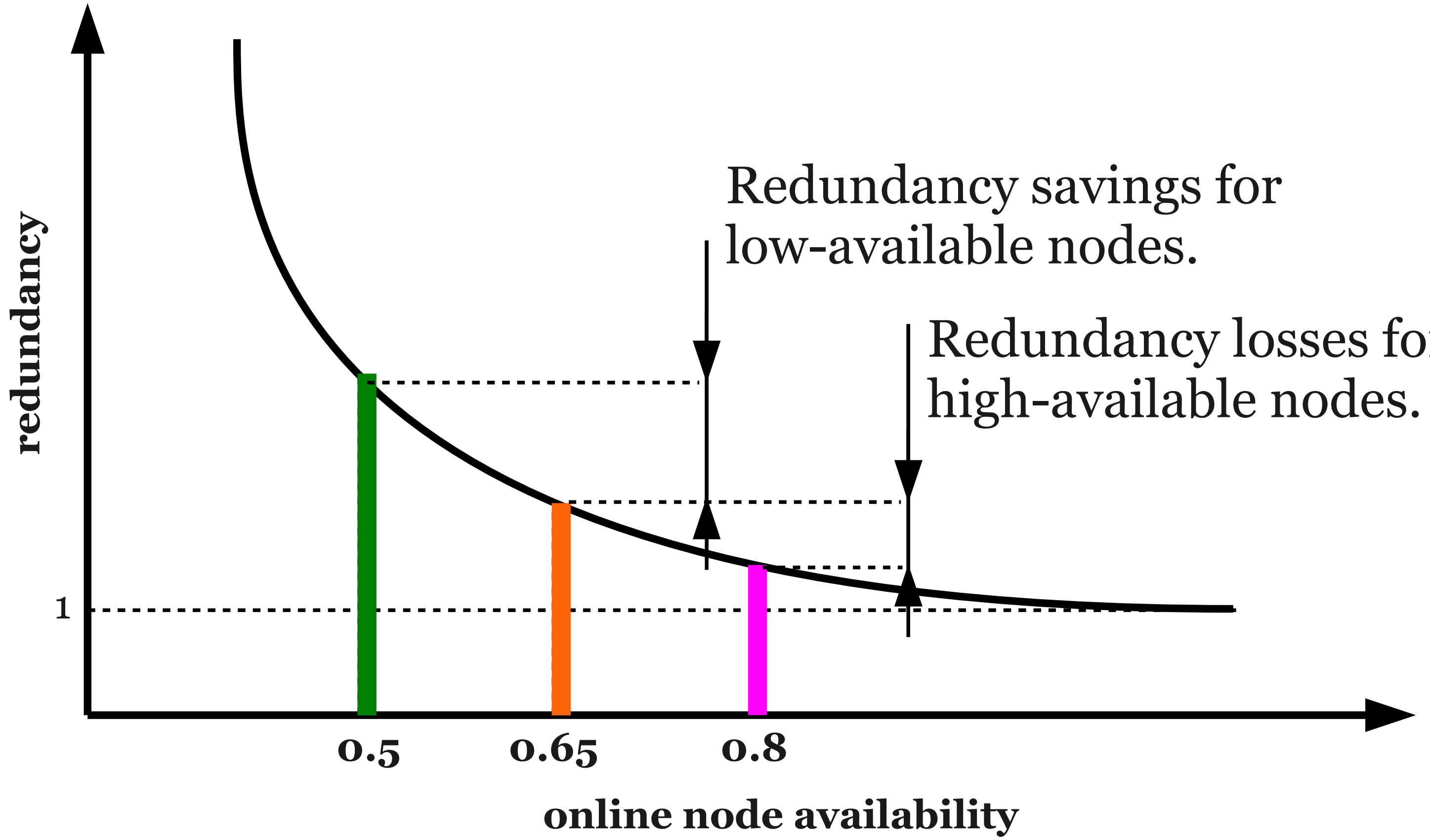
Interesting Observation



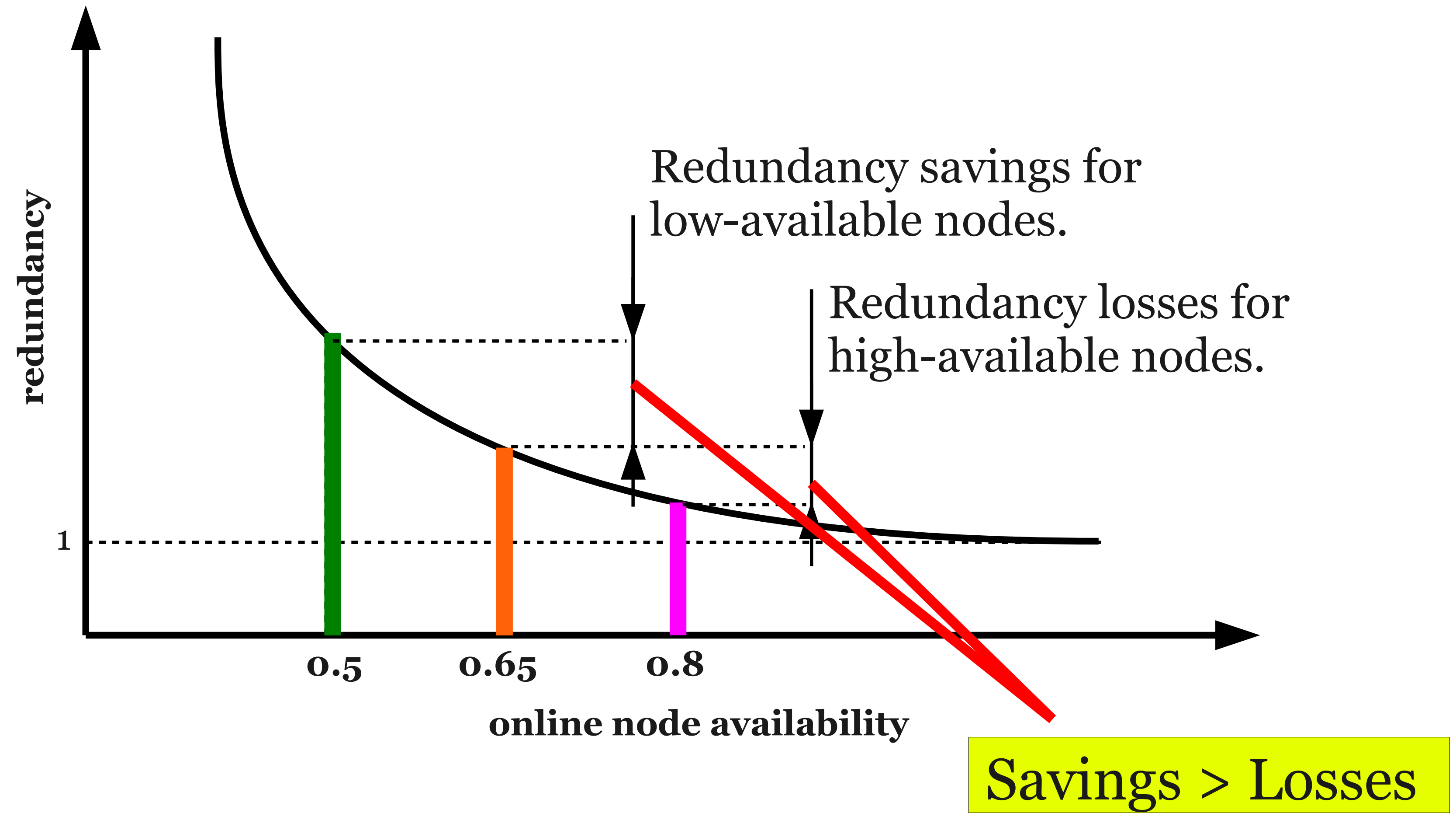
Interesting Observation



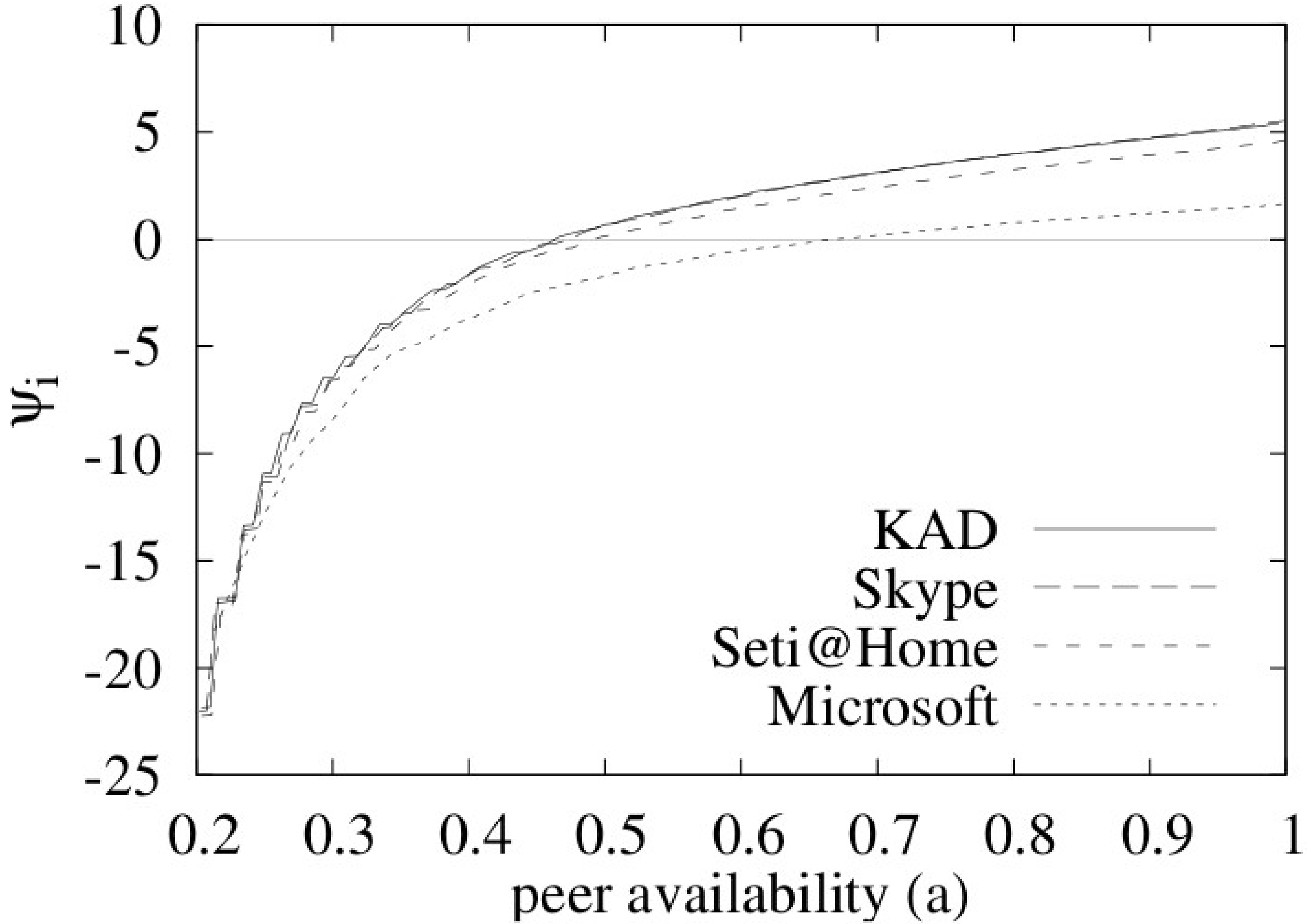
Interesting Observation



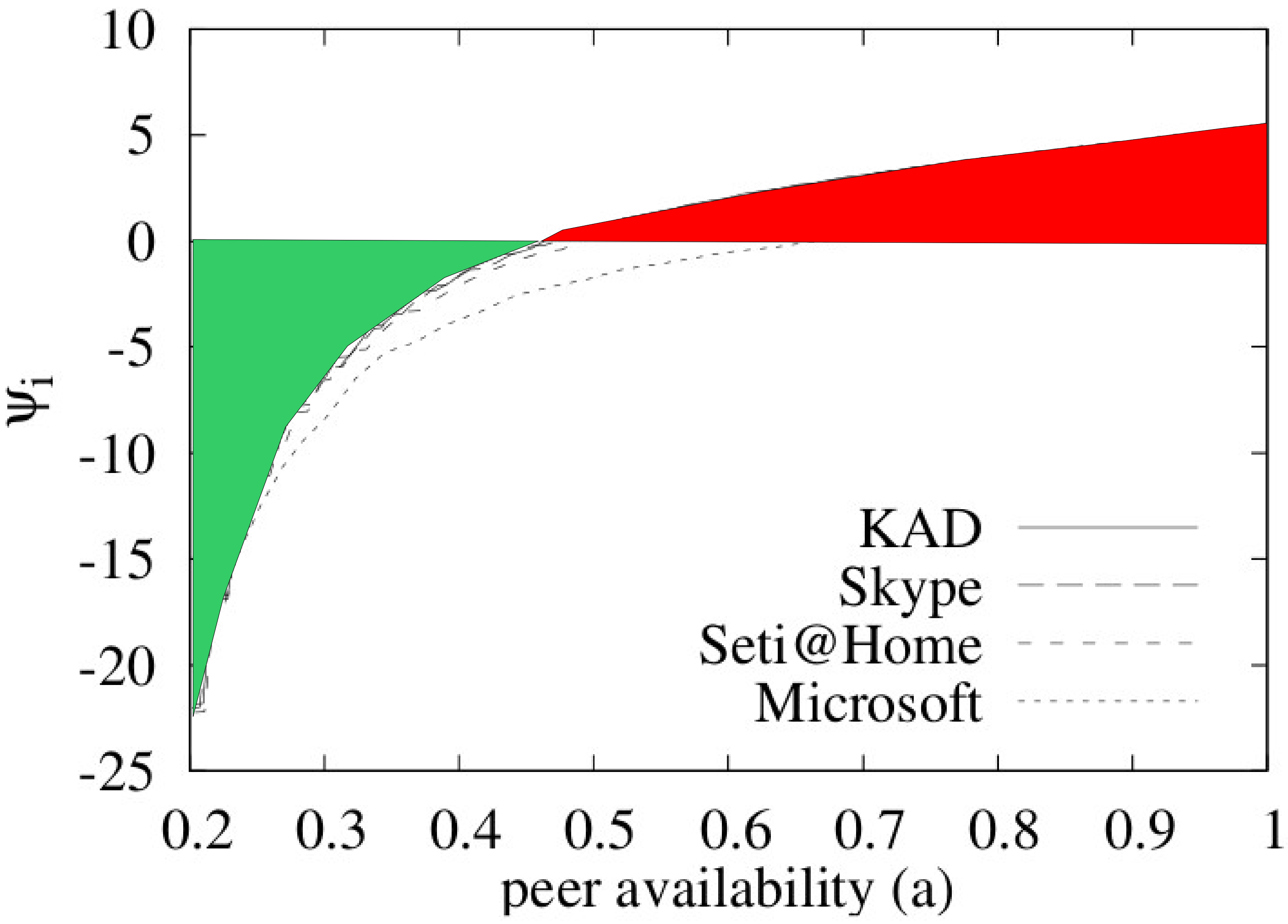
Interesting Observation



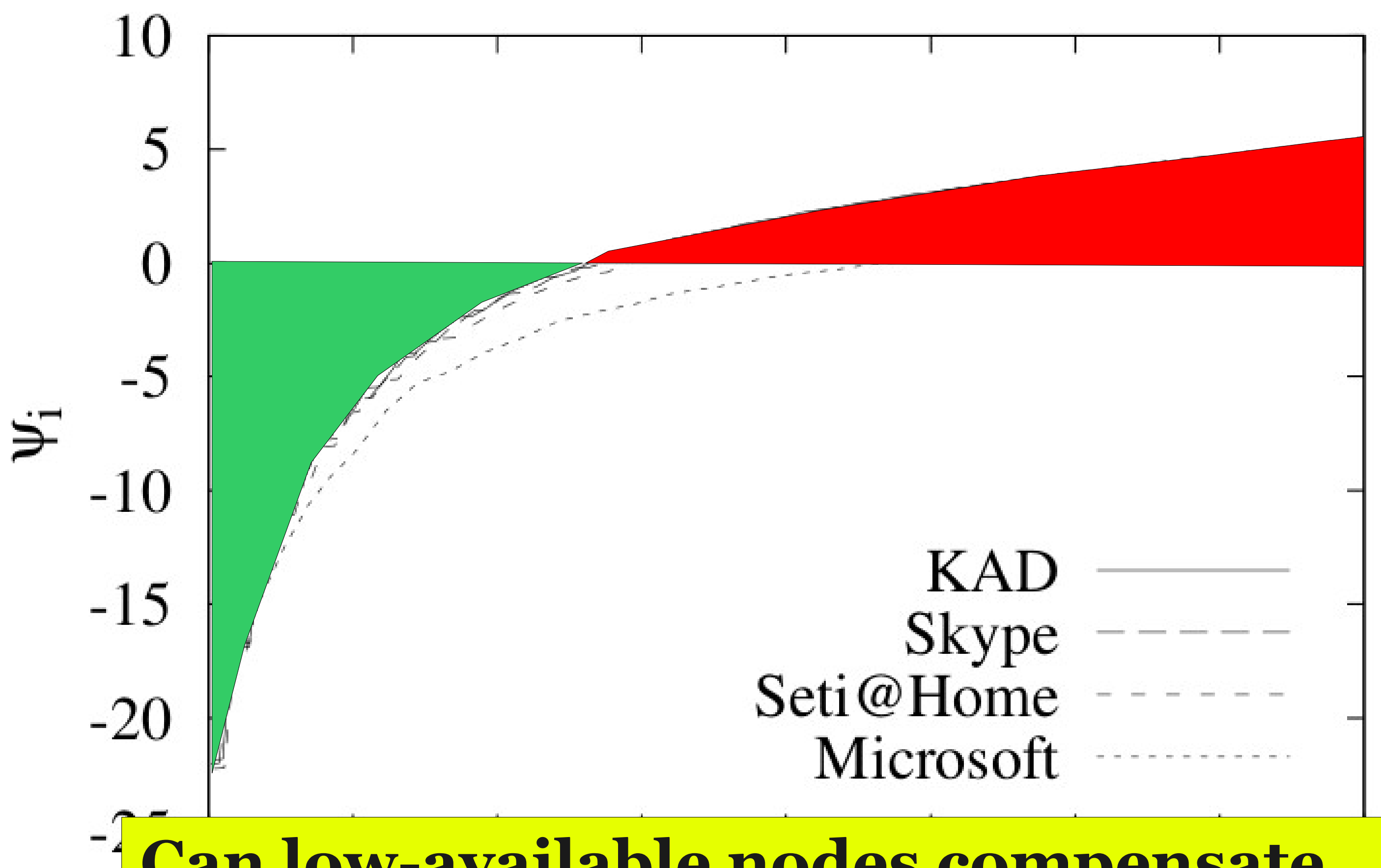
Interesting Observation



Interesting Observation

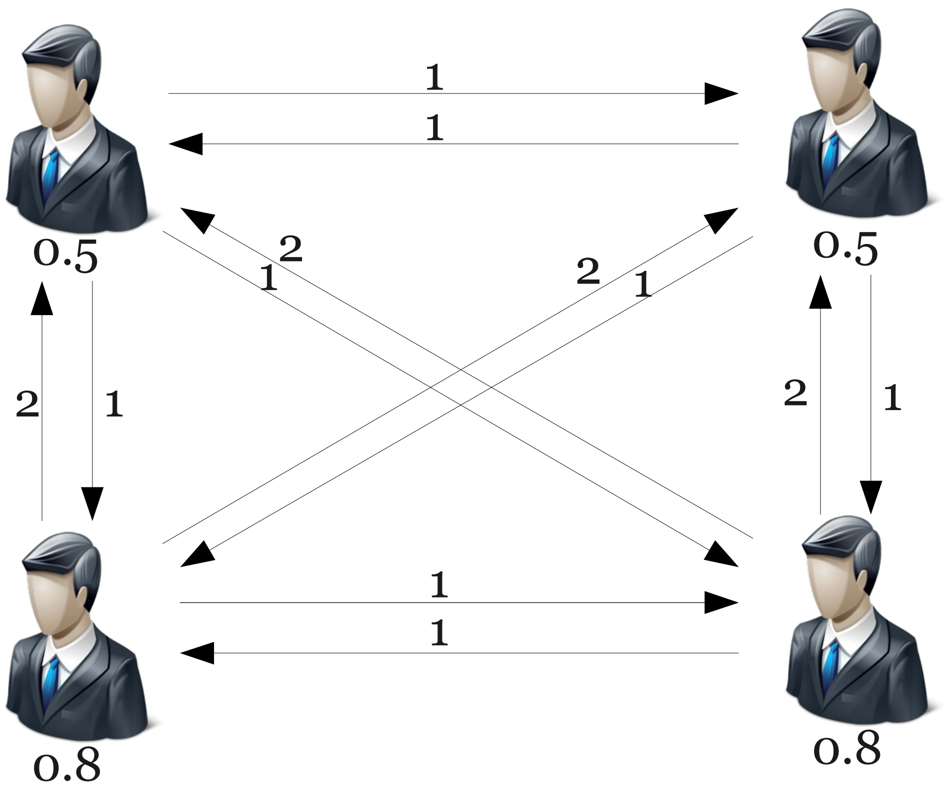


Interesting Observation

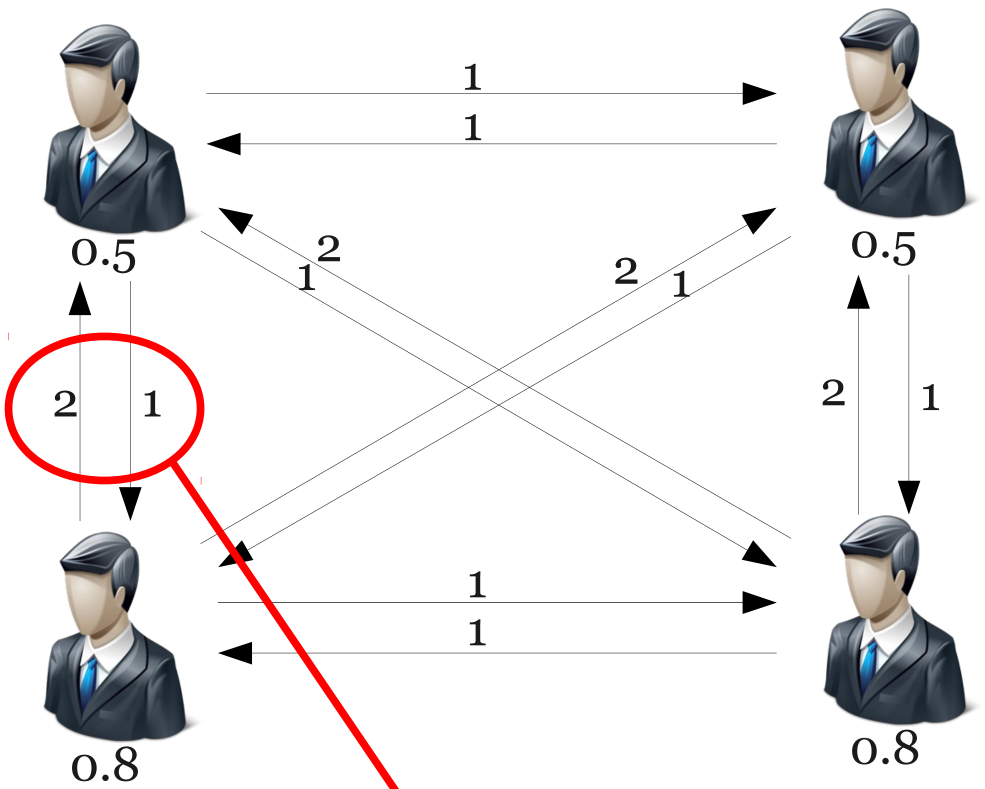


Can low-available nodes compensate the losses of high-available nodes, and globally reduce the amount of resources all users contribute ?

Asymmetric Reciprocal Exchanges



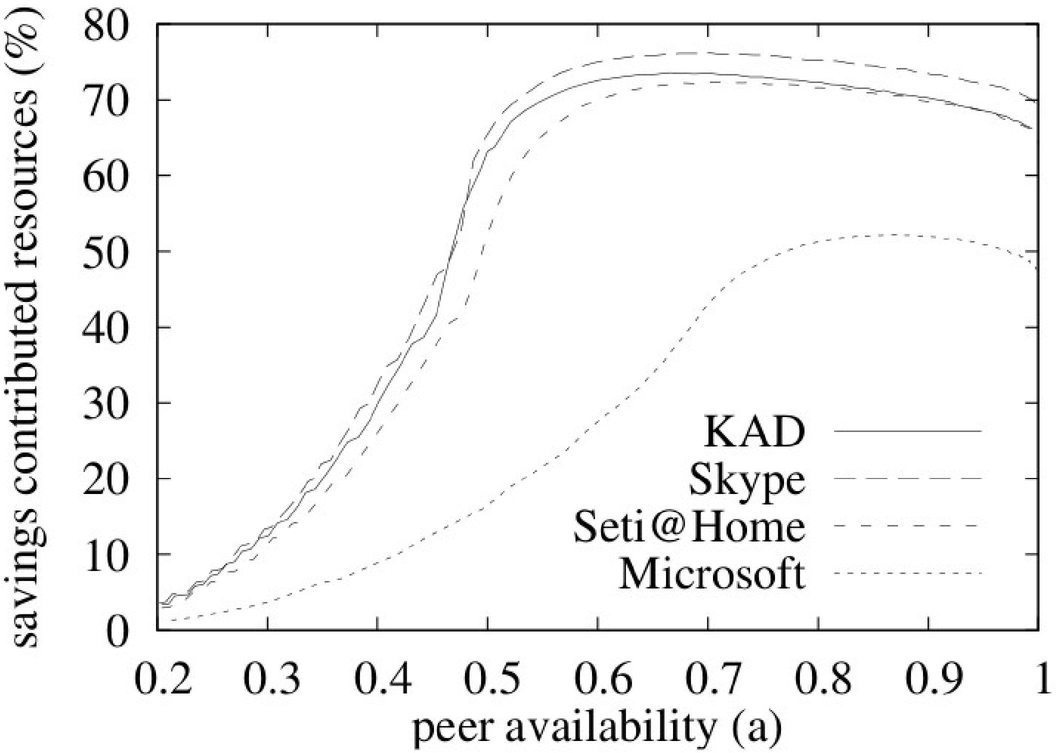
Asymmetric Reciprocal Exchanges



Given the availabilities of two users, which is the optimal asymmetric exchange ratio?

Our Implementation

- Solve a system of linear equations, defined by to proportionality rules:
  - Global savings are distributed proportional to the online availability of each peer.
  - Each peer compensates the partners more available than her proportionally to their online availability.



Outline

1. Introduction
2. Distributed Storage Allocation Problem
3. Homogeneous Distributed Systems
4. Heterogeneous Distributed Systems:
   1. Orchestrated Systems
   2. P2P Systems
5. **Other Open Problems in Distributed Storage Systems.**

Other Open Problems

- Repair Problem:
  - Large datacenters register 3-6% of hard drives failures every year → high repair communication
  - Repair redundant blocks without reconstructing the original file [1],[2].
- Allocation problems
  - Consider datacenter network topologies and different correlated failure patterns.
- Data access:
  - Replication guarantees efficient accesses (no decoding) and allows to move computation to where data is stored (less communication).
  - Improve data assignments in coding to minimize these inefficiencies.
- Data insertion:
  - In erasure codes data is inserted from a single node that has to generate and store n redundant blocks → low insertion throughput.
  - Use in-network coding to improve the data insertion throughput [3].

[1] Network Coding for Distributed Storage Systems. *Dimakis et al*. IEEE Transactions on Information Theory.

[2] Self-repairing Homomorphic Codes for Distributed Storage Systems. *Oggier and Datta*. Infocom 2010.

[3] In-Network Redundancy Generation for Opportunistic Speedup of Backup. *Pamies, Datta and Oggier*. 2011

Thanks

- Q&A