The EigenTrust Algorithm for Reputation Management in P2P Networks

Sepandar D. Kamvar

Mario T. Schlosser

Hector Garcia-Molina

Stanford University

P2P Networks

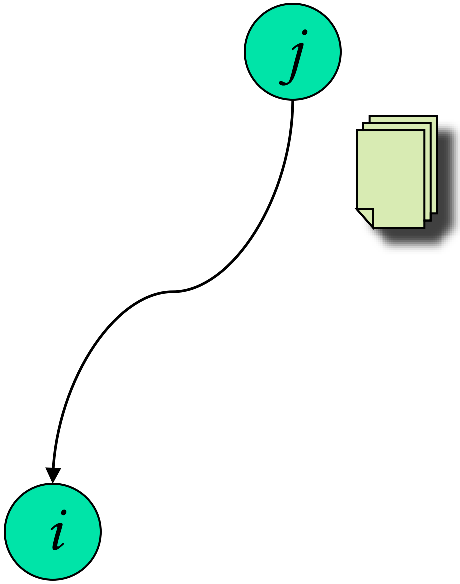
- Open and anonymous
  - Benefits
    - Robust, Scalable, Diverse
  - Problems
    - Malicious peers
    - Inauthentic files
      - Viruses/Malware
      - Tampered files

Identifying malicious peers is more pressing than identifying inauthentic files

Reputation Systems

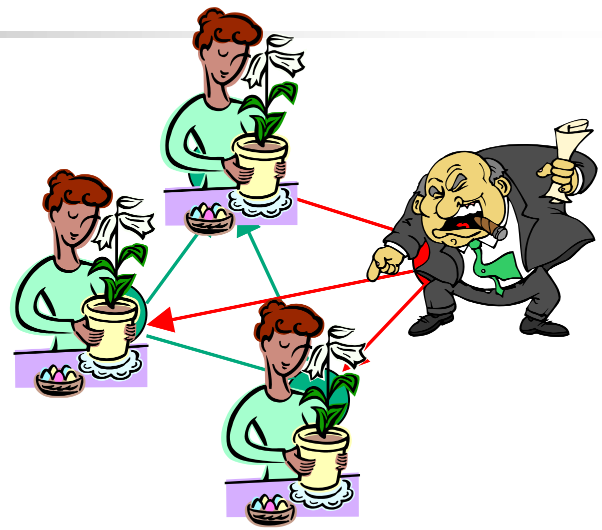
- Reputation Systems
  - *Global*: Centralized system (eBay)
  - *Local*: Distributed System

- *Key Idea of EigenTrust:* Each peer *i* is assigned a *global* trust value that reflects the *local* experiences of all the peers in the network



Problem

- **Problem:**
  - Reduce inauthentic files distributed by malicious peers on a P2P network.



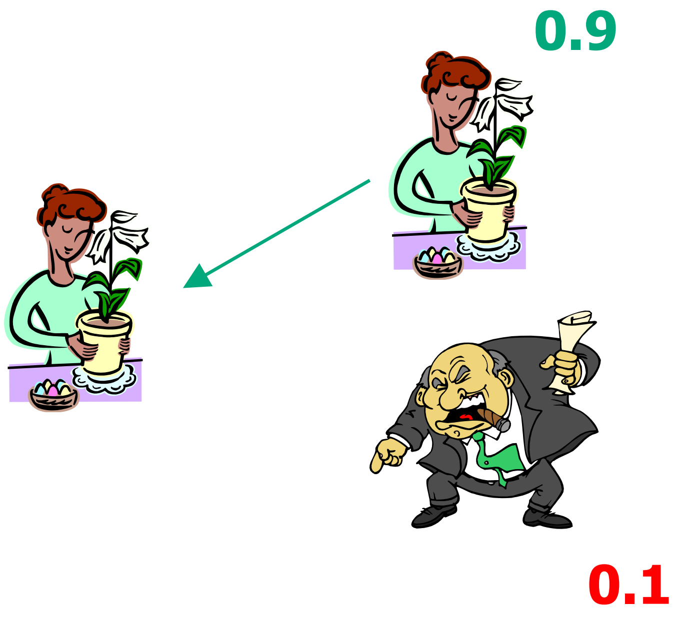
- **Motivation:**

"Major record labels have launched an aggressive new guerrilla assault on the underground music networks, flooding online swapping services with bogus copies of popular songs."

-Silicon Valley Weekly

Problem

- **Goal:** To identify sources of inauthentic files and bias peers against downloading from them.
- **Method:** Give each peer a *trust value* based on its previous behavior.



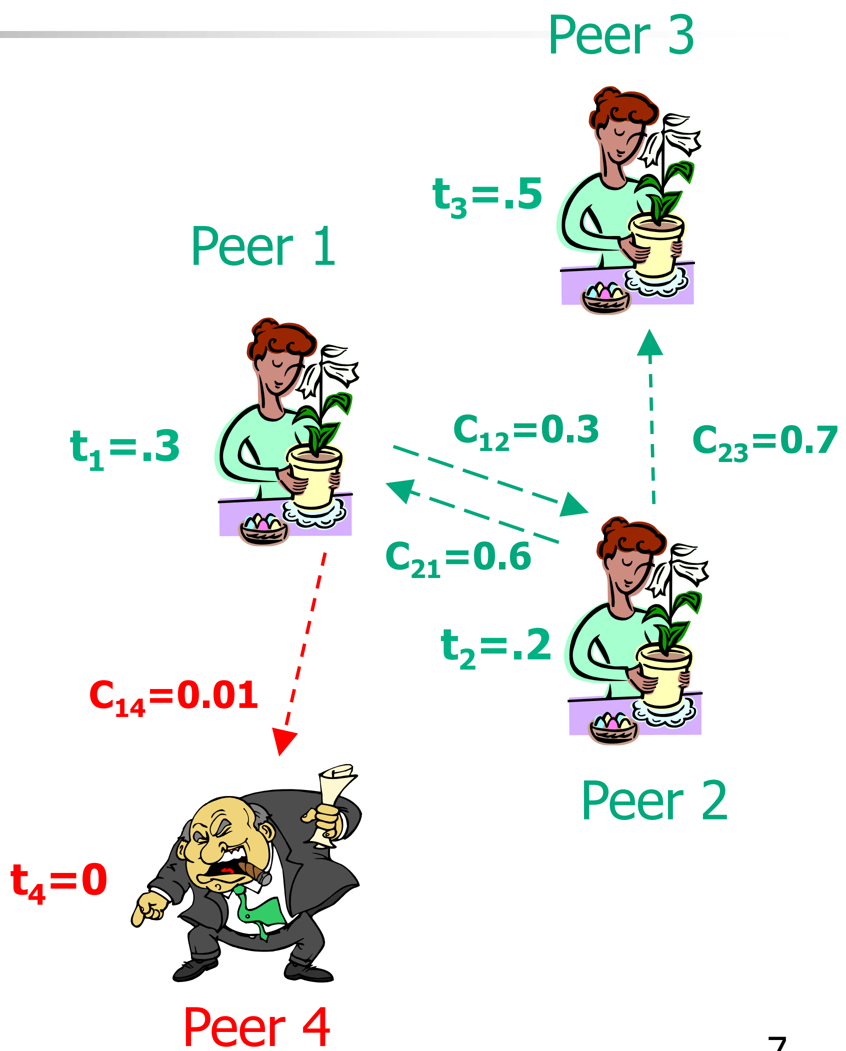
Some approaches

- Past History
- Friends of Friends
- EigenTrust

Terminology

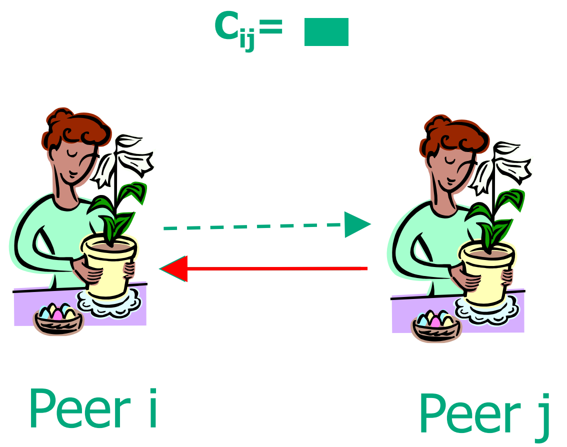
- **Local trust value: c_{ij}.** The opinion that peer i has of peer j, based on past experience.

- **Global trust value: t_i.** The trust that the entire system places in peer i.



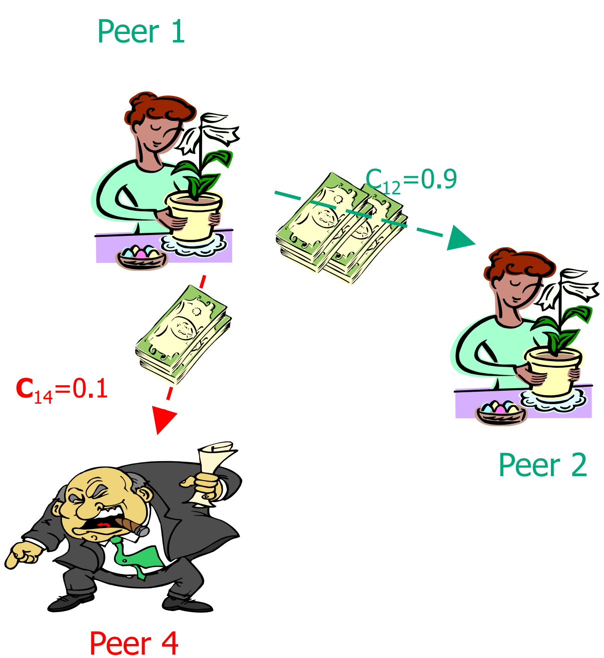
Local Trust Values

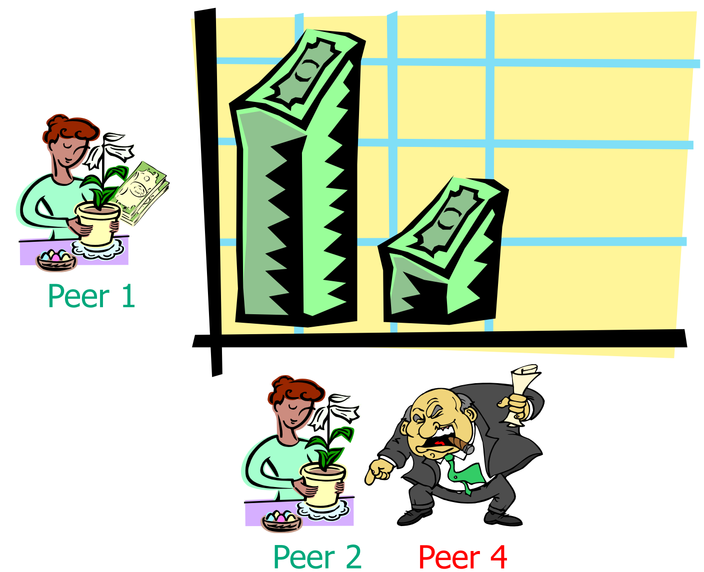
- Each time peer i downloads an authentic file from peer j, c_{ij} increases.
- Each time peer i downloads an inauthentic file from peer j, c_{ij} decreases.



Normalizing Local Trust Values

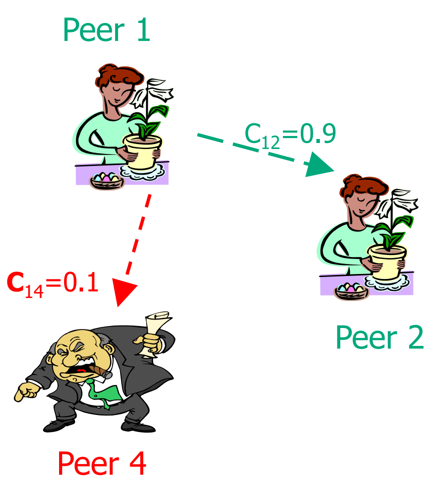
- All c_{ij} non-negative
- $c_{i1} + c_{i2} + \ldots + c_{in} = 1$

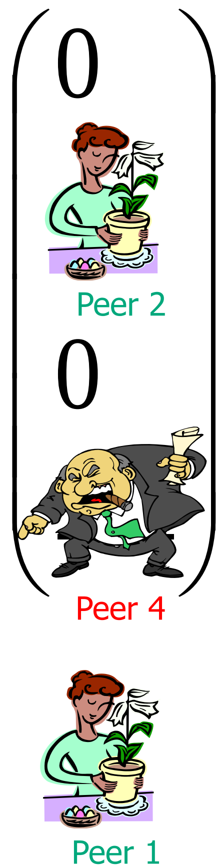




Local Trust Vector

- **Local trust vector c_i:** contains all local trust values c_{ij} that peer i has of other peers j.



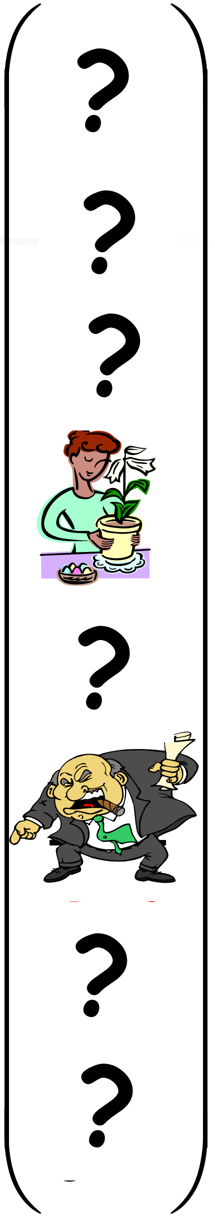


$$= \begin{pmatrix} 0 \\ 0.9 \\ 0 \\ 0.1 \end{pmatrix}$$

$\mathbf{c_1}$

Approach 1: Past history

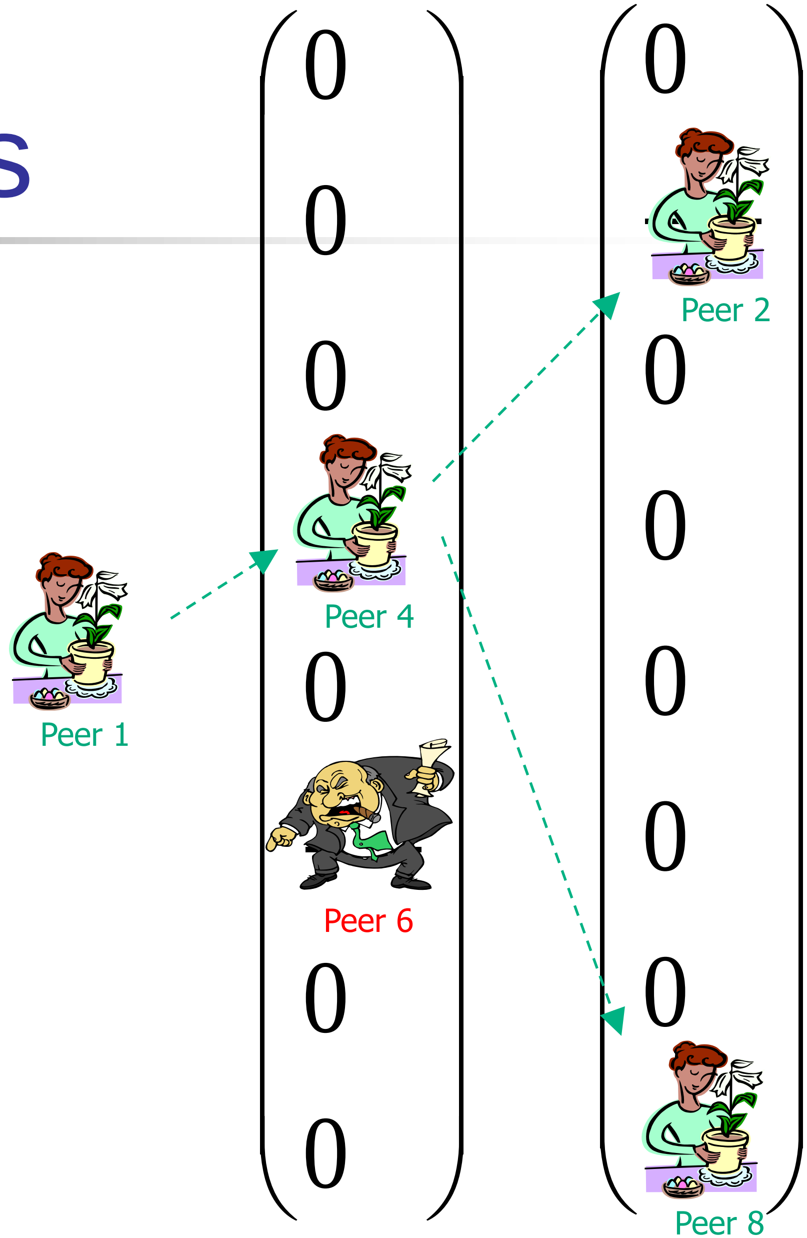
- Each peer biases its choice of downloads using its own opinion vector $\mathbf{c_i}$.
- If it has had good past experience with peer j, it will be more likely to download from that peer.
- **Problem**: Each peer has limited past experience. Knows few other peers.



Approach 2: Friends of Friends

- Ask for the opinions of the people who you trust.

(Cf. Referral trust)

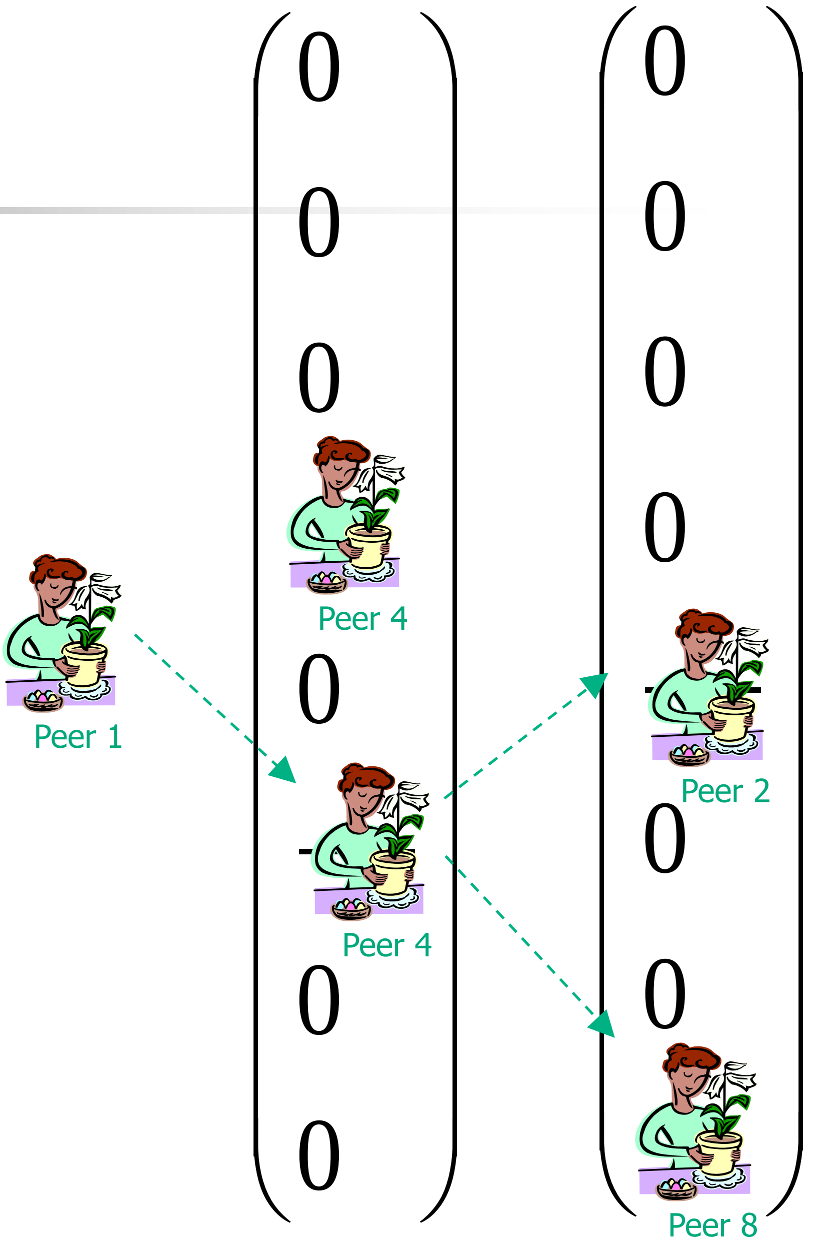


Friends of Friends

- Weight their opinions by your trust in them.

(Cf. Referral trust = Functional trust)

(Cf. Transitivity)



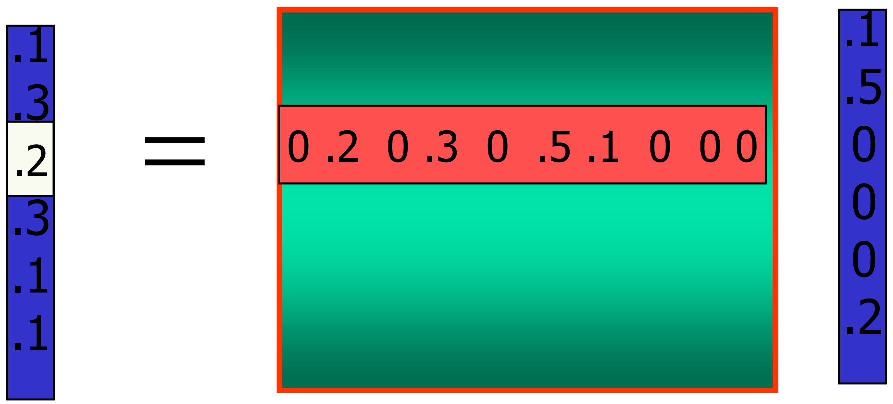
The Math : Transitive Trust

$$c'_{ik} = \sum_j c_{ij} \cdot c_{jk}$$

What they think of peer k.

Ask your friends j

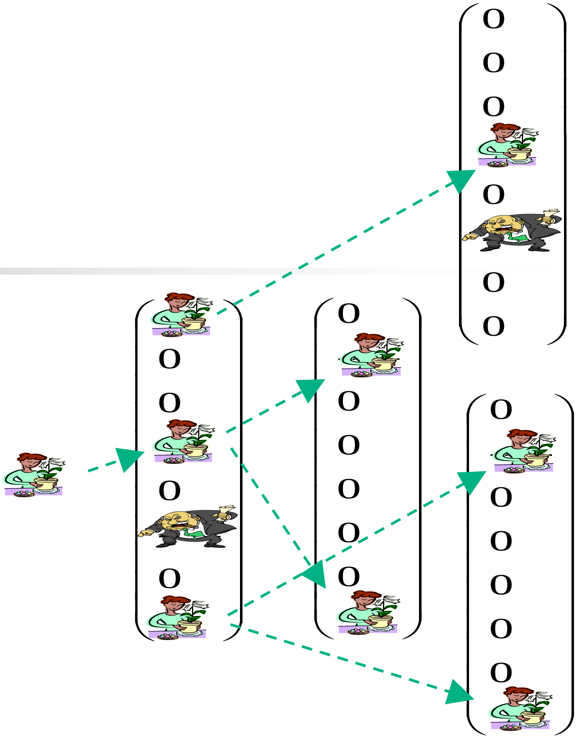
And weight each friend's opinion by how much you trust him.

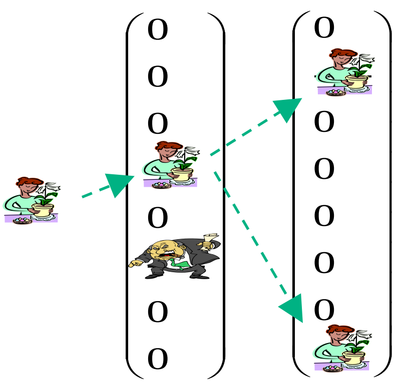


$$\mathbf{c}'_{\mathbf{i}} = \mathbf{C}^{\mathbf{T}} \mathbf{c}_{\mathbf{i}}$$

Problem with Friends

- Either you know a lot of friends, in which case, you have to communicate, compute and store many values.
- Or, you have few friends, in which case you won't know many peers, even after asking your friends.



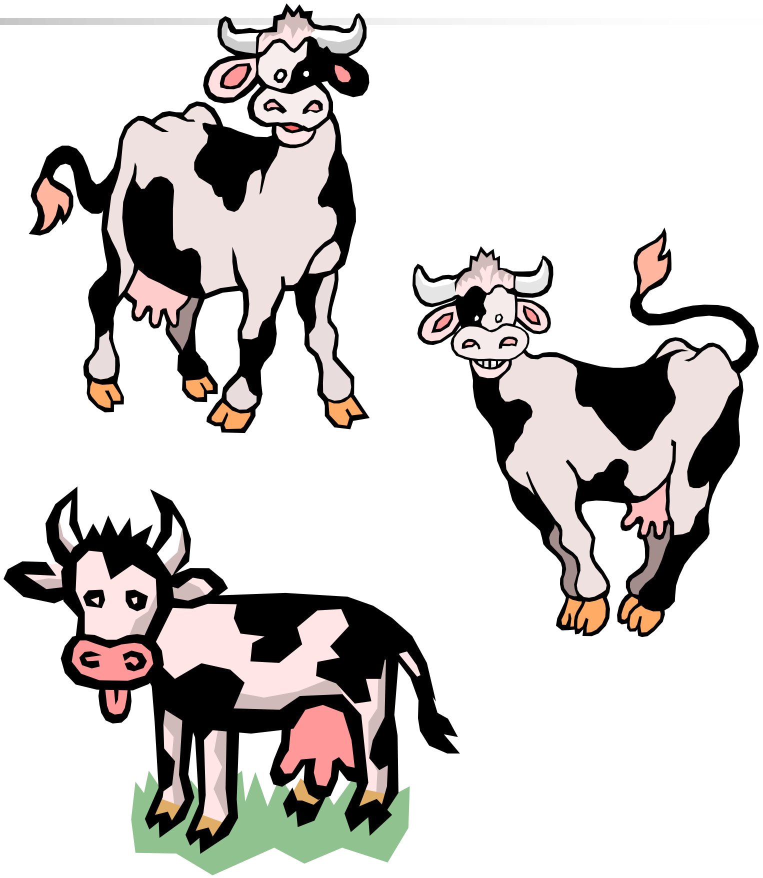


Eigen Trust: Dual Goal

- We want each peer to:
  - Know all peers.
  - Perform minimal computation (and storage).

Knowing All Peers

- Ask your friends: $t = C^T c_i$.
- Ask their friends: $t = (C^T)^2 c_i$.
- Keep asking until the cows come home: $t = (C^T)^n c_i$.



Minimal Computation

- Luckily, the *trust vector* **t**, if computed in this manner, converges to the same thing for every peer!
- Therefore, each peer doesn't have to store and compute its own trust vector. The whole network can cooperate to store and compute **t**.

Non-distributed Algorithm

- Initialize:

$$\mathbf{t}^{(0)} = \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}^{\mathbf{T}}$$

- Repeat until convergence:

$$\mathbf{t}^{(\mathbf{k}+1)} = \mathbf{C}^{\mathrm{T}}\mathbf{t}^{(\mathbf{k})}$$

Basic EigenTrust Algorithm

- Assumption: include central server at this stage
  - A server stores all the c_{ij} values and performs the computation

$\vec{t}^{(0)} = \vec{e};$ $\quad e_i = 1/n$

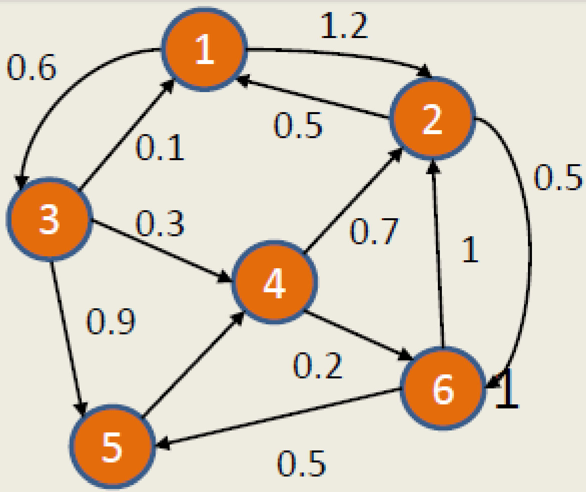
repeat

$\quad \vec{t}^{(k+1)} = C^T \vec{t}^{(k)};$

$\quad \delta = ||t^{(k+1)} - t^k||;$

until $\delta < \epsilon;$

An Illustration Example of EigenTrust



Normalization

$$C = \begin{matrix} 0 & 0.67 & 0.33 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0.5 \\ 0.08 & 0 & 0 & 0.23 & 0.69 & 0 \\ 0 & 0.78 & 0 & 0 & 0 & 0.22 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.67 & 0 & 0 & 0.33 & 0 \end{matrix}$$

$$t^0 = \begin{matrix} 0.1667 \\ 0.1667 \\ 0.1667 \\ 0.1667 \\ 0.1667 \\ 0.1667 \end{matrix} \qquad t^1 = C^T t^0 = \begin{matrix} 0.0967 \\ 0.3534 \\ 0.0550 \\ 0.2050 \\ 0.1700 \\ 0.1200 \end{matrix} \qquad t^2 = C^T t^1 = \begin{matrix} 0.1811 \\ 0.3051 \\ 0.0319 \\ 0.1827 \\ 0.0776 \\ 0.2218 \end{matrix} \quad \ldots\ldots\ldots \quad \begin{matrix} 0.1764 \\ 0.3434 \\ 0.0582 \\ 0.1188 \\ 0.1055 \\ 0.1979 \end{matrix}$$

Conclusion

- Eigentrust
  - Dramatically reduces number of inauthentic files on the network.
  - Robust to malicious peers.
  - Low overhead.
- Paper available at http://www.stanford.edu/~sdkamvar/research.html