The EigenTrust Algorithm for Reputation Management in P2P Networks

S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina

Presented by Jay Liu for CS395T

2/27/2009

THE UNIVERSITY OF TEXAS AT AUSTIN
WHAT STARTS HERE CHANGES THE WORLD

Reputation and trust

- Reputation-based trust system uses knowledge from earlier interactions to assess trust. (Aberer and Despotovic '01)
- Information is local: peer i knows earlier transactions with peer j.
- Make that data globally usable: all peers know that i remembers j.
  - Cooperative

Centralized reputation-based trust – Amazon.com

Motivation for P2P

- P2P is very popular
- There are malicious users
- Malicious users share inauthentic files
  - Decoy/Incorrect ID
  - Virus
- Discourages honest users
- No centralized storage + computing

Decoy (WSJ 10/18/2006)

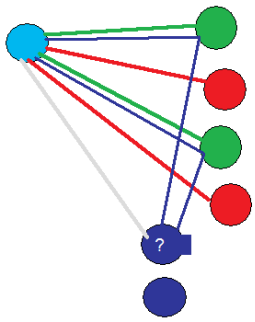
- "Record labels ... plant "decoy," or fake, files on the sites … for as many as 30 of the top 100 Billboard songs..."
- "By inserting promotional material into the decoy files… record labels and other marketers can turn what is now an antipiracy tool into an advertising medium."

Source: http://online.wsj.com/public/article/SB116113611429796022-_5EZVscJYWWFqv1AmPvXCiOjJms_20071018.html?mod=blogs

Design principles

- Self-policing: no central authority
- Anonymity: no identification of peer
- Newcomers start with no trust
- Low overhead, complexity
- Prevent collusion

Basic design



Transaction *tr*

- Transfer to i from j
- Transaction tr_{ij} is binary
  - 1 if good
  - -1 if not good

Satisfaction

- Satisfaction:

$$s_{ij} = \sum tr_{ij}$$

- s_{ij} is
  - = 0 if user is new
  - < 0 if malicious

Normalized local trust value c_{ij}

- A peer may consistently rate others highly: normalize
- Takes max(s_{ij} , 0)
  - Both new and malicious appear to be 0!

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$$

Inferring trust – Reputation

- i knows j
- i gets to know k through j

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

- Transitivity:
  - i gets to know l through j, k, etc…

Matrix notation

- This can be formulated as matrix

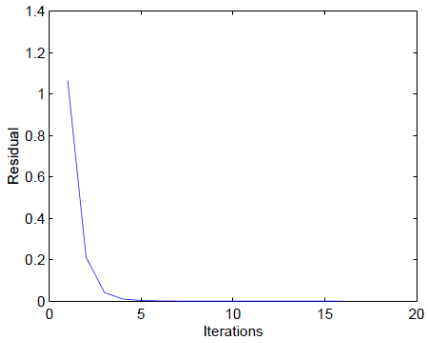
$$t = (C^T)^n c_i$$

$$\begin{bmatrix} t_{ik_1} \\ t_{ik_2} \end{bmatrix} = \begin{bmatrix} c_{j_1k_1} & c_{j_1k_2} \\ c_{j_2k_1} & c_{j_2k_2} \end{bmatrix}^T \begin{bmatrix} c_{ij_1} \\ c_{ij_2} \end{bmatrix}$$

EigenTrust

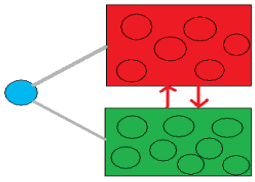
- t_{ik} of node k converges to same value for all peers i
  - Consistent global view of peer k
- Specifically, left principal eigenvector of C
  - Hence "EigenTrust"

Convergence rate



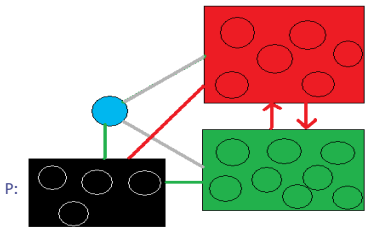
Peer types

- A malicious peer may only award trust to other malicious peers
- Solution: Not all peers are equal



Pre-trusted peers

- P peers are always trusted
  - Their existence is a basic design requirement



Pre-trusted peers (cont'd)

- P needs to be chosen carefully
  - Penetration of P by malicious user will significantly impact system
- Each such peer j is trusted by $p_j = 1/|P|$
  - Non-pre-trusted peers have $p = 0$
- P needs to be active.
  - They need to participate in trust building.

Benefit to new/inactive peer

- For a peer who is:
  - new, or
  - does not participate in trust building, or
  - cannot find anyone trustworthy
- sum(s_{ij}) = 0, then $c_{ij} = p_j$
  - I.e. only trust pre-trusted peers

EigenTrust algorithm

- Find t as before, but always place a% of trust in pre-trusted peers

$\vec{t}^{(0)} = \vec{p};$

repeat

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

$\quad \vec{t}^{(k+1)} = (1-a)\vec{t}^{(k+1)} + a\vec{p};$

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

until $\delta < \epsilon;$

EigenTrust computation

- EigenTrust values need to be computed and stored. 3 suggestions:

1. Each peer computes and stores own values
   - Peer could be untrustworthy and lie
2. Pre-trusted peers compute and store all values
   - Still requires central bottlenecks
3. Peer computes and stores other peer(s)'s values
   - Requires security

If peers are honest…

- Reliable peer can store both its local trust value c and its global trust value t
- A_i: set of peers which downloaded from i
  - A_i provides trust value for i
- B_i: set of peers from which i downloaded
  - i provides trust value for B_i

If peers are honest… (cont'd)

- Each peer knows its own p value.
  - Pre-trusted peer recognizes itself.
- Each peer *in practice* has few neighbor peers, so:
  - $|A_i|$ and $|B_i|$ are small
  - Few computation terms.
  - Few message exchanges

Distributed EigenTrust Algorithm

Each peer i do {
Query all peers $j \in A_i$ for $t_j^{(0)} = p_j$;
repeat
Compute $t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + c_{2i}t_2^{(k)} + \ldots + c_{ni}t_n^{(k)}) + ap_i$;
Send $c_{ij}t_i^{(k+1)}$ to all peers $j \in B_i$;
Compute $\delta = |t_i^{(k+1)} - t_i^{(k)}|$;
Wait for all peers $j \in A_i$ to return $c_{ji}t_j^{(k+1)}$;
until $\delta < \epsilon$.;
}

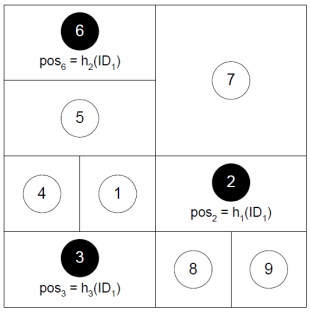
Security – Unreliable peers

- Peers cannot be trusted with providing own trust values. (Score managers)
- Single peer cannot be trusted with providing trust value for another peer.
  - … also for redundancy considerations.
- Need multiple peers to agree on a trust value for a given peer.
- Need scoring managers to be random.

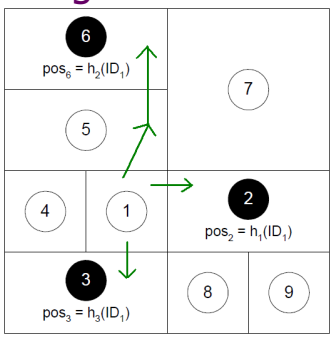
Finding Scoring Managers

- Map peers using DHT, e.g. CAN
- Input: Peer identifier
  - Hash of IP, TCP port
- Output: Coordinate space
  - A region in CAN managed by peer(s)
- Out of scope issues: vote, replication, join/leave, churn…

Content-Addressable Network (CAN)



CAN Routing



Secure EigenTrust algorithm

foreach *peer* i **do**

Submit local trust values $\vec{c_i}$ to all score managers at positions $h_m(pos_i), m = 1 \ldots M-1$;

Collect local trust values $\vec{c_d}$ and sets of acquaintances B_d^i of daughter peers $d \in D_i$;

Submit daughter d's local trust values c_{dj} to score managers $h_m(pos_d), m = 1 \ldots M-1, \forall j \in B_d^i$;

Collect acquaintances A_d^i of daughter peers;

Secure EigenTrust algorithm (cont'd)

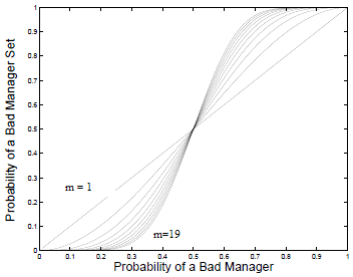
foreach *daughter peer* $d \in D_i$ **do**
  Query all peers $j \in A_d^i$ for $c_{jd}p_j$;
  repeat
    Compute $t_d^{(k+1)} = (1-a)(c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \ldots + c_{nd}t_n^{(k)}) + ap_d$;
    Send $c_{dj}t_d^{(k+1)}$ to all peers $j \in B_d^i$;
    Wait for all peers $j \in A_i^d$ to return $c_{jd}t_j^{(k+1)}$;
  until $|t_d^{(k+1)} - t_d^{(k)}| < \epsilon$.;
end
end

Secure algorithm notes

- Say, i manages score for j
- i does not know identity of j, only the hash value of j's ID
  - Maintains anonymity
  - But i may be in B_j …! So it can guess…
- Question: who tracks p_j?

Aside: Malicious score manager (Kamvar et al, 2004)

- E.g. always return 0 score

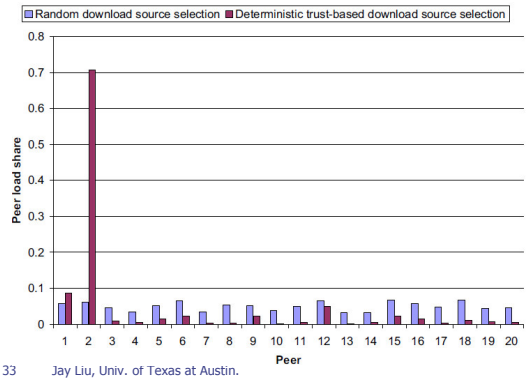


Kamvar, Sepandar and Yang, Beverly and Garcia-Molina, Hector (2004) *Secure Score Management for Peer-to-Peer Systems.* Technical Report, Stanford.

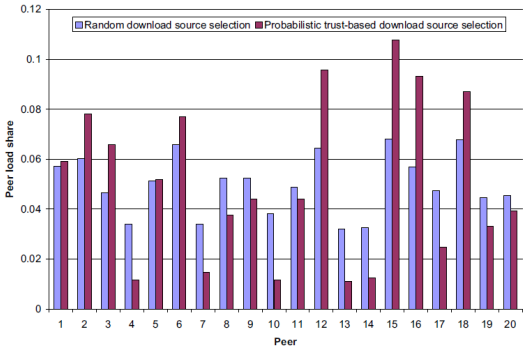
Using the Result: Peer Selection

1. **Deterministic**: always pick most reputable source.
   - Will cause overload and over reliance.
   - Create peak trust peer.
   - New peers cannot compete
2. **Probabilistic**: Regard trust value as the probability of picking the peer.
   - If peer has 0 value, use 10% (!)
     - But ideally, malicious should get 0%.

Deterministic Peer Selection



Probabilistic Peer Selection



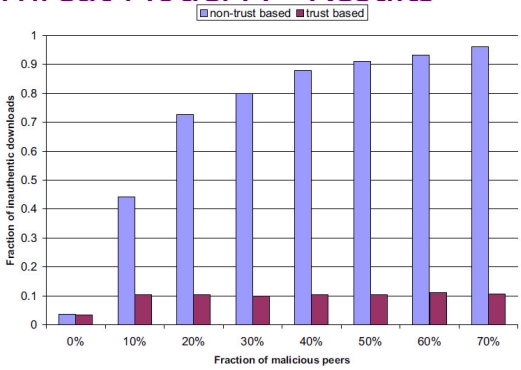
Using the Result – Incentives

- Reward good behavior
- Those with high trust get higher priority during download
- Building trust requires active and honest participation
  - Remove bad files
  - Rate other users

Threat Model A – Individual malicious user

- **Always** upload inauthentic files
- Malicious peers act independently from each other
- Reward other malicious peers who provide inauthentic files when found

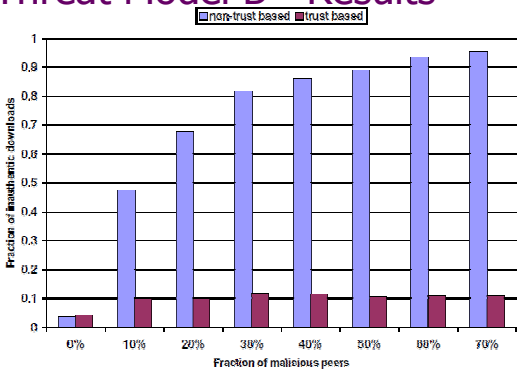
Threat Model A - Results



Threat Model B - Collective

- **Always** upload inauthentic files
- Malicious peers are cooperating
- Malicious peers know each other, express high degrees of trust in each other
- Attempts to lead user to believe all such peers are trustworthy

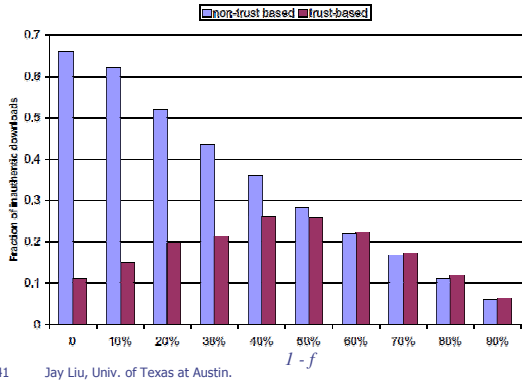
Threat Model B - Results



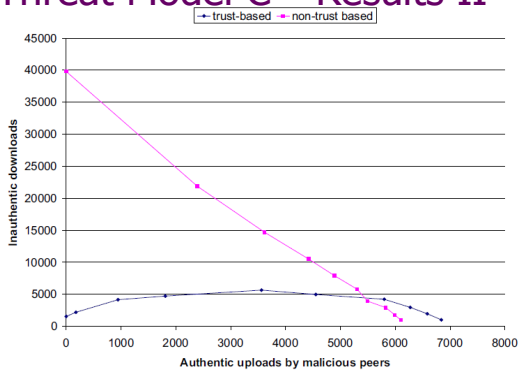
Threat Model C – Camouflage

- Provides **inauthentic** files in f **%** of all requests.
  - Provides authentic files in 1 - f % of all cases. This is a cost.
- Forms a malicious collective
- Some peers have positive trust value.
- Maximum effectiveness of attack at f = 50%.
  - f > 50%: trust eventually disappears
  - f < 50%: high cost to maintain attack. Decreasing marginal returns.

Threat Model C – Results



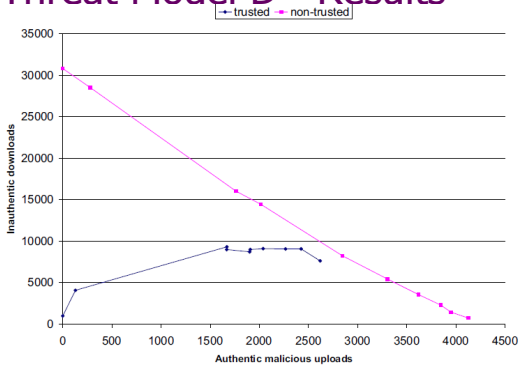
Threat Model C – Results II



Threat Model D - Spies

- Collective divided into two roles.
- **Type D nodes** provide authentic files
  - Appear good. Gain positive trust.
- **Type B nodes** are always malicious.
- Type D assigns high trust in Type B.
- Effective attack

Threat Model D - Results



Briefly: Threat Model E – "Sybil"

- A flood of malicious users in a collective
- Only effective where join and leave have low cost
  - so, increase cost of entry

Briefly: Threat Model F – Virus Disseminator

- **Very rarely** provides inauthentic file.
  - Yet, such file may be highly damaging. (E.g. virus.)
- Malicious, or user error? Distrust or forgive?
- Difficult to detect. Not solved.

Conclusion

- EigenTrust combats malicious peers by significantly reducing incidence rate, or increasing the cost of attack.
- Secure EigenTrust relies on DHT