Reputation Management in P2P Networks

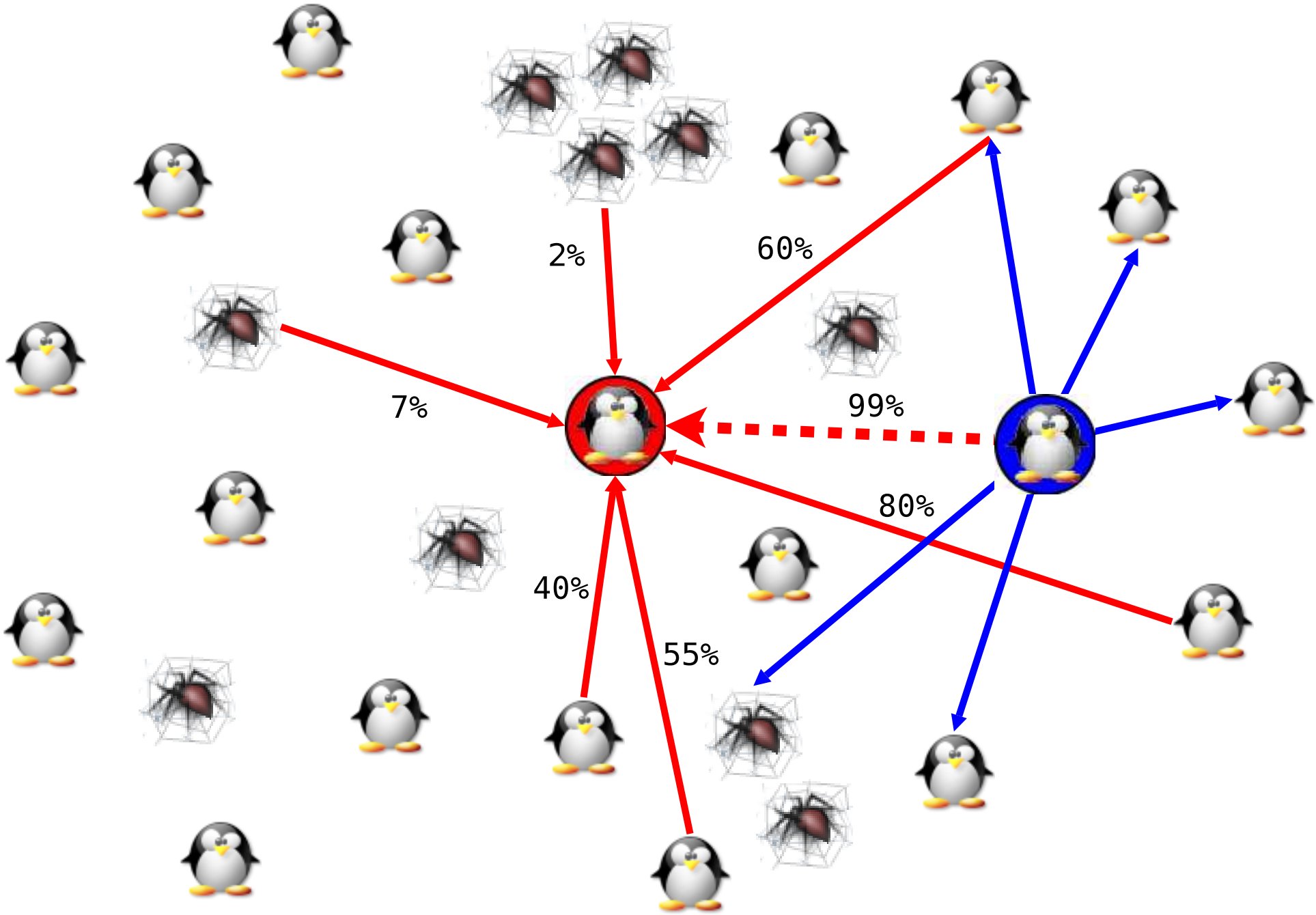
– The EigenTrust Algorithm –

presented by

Adrian Alexa

supervised by

Anja Theobald


2%
60%
7%
99%
80%
40%
55%

- General Overview
- Local and Global Trust Values
- From Basic to Distributed EigenTrust
- What about Security ?
- Results and Conclusions

- **General Overview**
- Local and Global Trust Values
- From Basic to Distributed EigenTrust
- What about Security ?
- Results and Conclusions

Problem:

- minimize the number of *inauthentic files* spread by malicious peers in a P2P System

Problem:

- minimize the number of *inauthentic files* spread by malicious peers in a P2P System

Goal:

- identify sources of inauthentic files
- bias peers belief when downloading from them

Problem:

- minimize the number of *inauthentic files* spread by malicious peers in a P2P System

Goal:

- identify sources of inauthentic files
- bias peers belief when downloading from them

Method:

- assign peers *global trust values* based on previous peer's behavior

Examples:

- ebay
- amazon, bizrate

- **Examples:**
  - ebay
  - amazon, bizrate
- **Reputation:** a measure obtained from earlier transactions

- **Examples:**
  - ebay
  - amazon, bizrate
- **Reputation:** a measure obtained from earlier transactions
- **Trust Management:** a mechanism that allows to establish reciprocal trust

Why Global Trust Values ?

- **GTV** represent the trust that all peers put in you; **it is unique**

Why Global Trust Values ?

- **GTV** represent the trust that all peers put in you; **it is unique**
- **Isolating Malicious Peers**
  - download from the most highly trusted peer
  - select peers from whom to download based on the distribution induced by peer's trust value
  - combine local trust with the global trust value:

$$t = at_{\text{global}} + (1 - a)t_{\text{local}}$$

Why Global Trust Values ?

- **GTV** represent the trust that all peers put in you; **it is unique**
- **Isolating Malicious Peers**
  - download from the most highly trusted peer
  - select peers from whom to download based on the distribution induced by peer's trust value
  - combine local trust with the global trust value:

$$t = at_{\text{global}} + (1 - a)t_{\text{local}}$$

- **Incent Peers to Share Files**
  - high trusted peers can be rewarded
  - reduce the numbers of free-riders

Bring order to chaos

- **anonymity:** peer's reputation should be associated with an *opaque* ID (usernames vs. IP address)

Bring order to chaos

- **anonymity:** peer's reputation should be associated with an *opaque* ID (usernames vs. IP address)
- **self-policing:** there is no central authority

Bring order to chaos

- **anonymity:** peer's reputation should be associated with an *opaque* ID (usernames vs. IP address)
- **self-policing:** there is no central authority
- **minimal overhead:** peers should not spend too much time and bandwidth for computing their reputation

Bring order to chaos

- **anonymity:** peer's reputation should be associated with an *opaque* ID (usernames vs. IP address)
- **self-policing:** there is no central authority
- **minimal overhead:** peers should not spend too much time and bandwidth for computing their reputation
- **robust to malicious collectives:** the system will try to isolate collectivities who want to subvert it

Bring order to chaos

- **anonymity:** peer's reputation should be associated with an *opaque* ID (usernames vs. IP address)
- **self-policing:** there is no central authority
- **minimal overhead:** peers should not spend too much time and bandwidth for computing their reputation
- **robust to malicious collectives:** the system will try to isolate collectivities who want to subvert it
- **newcomers are not privileged:** malicious peers can't gain anything by changing their identity.

- General Overview
- **Local and Global Trust Values**
- From Basic to Distributed EigenTrust
- What about Security ?
- Results and Conclusions

- peer i download a file form peer j
  - $sat(i,j) =$ the number of satisfactory transaction
  - $unsat(i,j) =$ the number of unsatisfactory transaction

- peer i download a file form peer j
  - $sat(i,j) =$ the number of satisfactory transaction
  - $unsat(i,j) =$ the number of unsatisfactory transaction
  - we can define the Local Trust Value as:

$$s_{ij} = sat(i,j) - unsat(i,j)$$

- peer i download a file form peer j
  - $sat(i,j) =$ the number of satisfactory transaction
  - $unsat(i,j) =$ the number of unsatisfactory transaction
  - we can define the Local Trust Value as:

$$s_{ij} = sat(i,j) - unsat(i,j)$$

- Problem: how to aggregate these local scores ?

- peer i download a file form peer j
  - $sat(i,j) =$ the number of satisfactory transaction
  - $unsat(i,j) =$ the number of unsatisfactory transaction
  - we can define the Local Trust Value as:

$$s_{ij} = sat(i,j) - unsat(i,j)$$

- Problem: how to aggregate these local scores ?
- normalize these values to obtain probabilities

$$0 \quad \leq \quad c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)} \quad \leq \quad 1$$

- peer i download a file form peer j
  - $sat(i,j) =$ the number of satisfactory transaction
  - $unsat(i,j) =$ the number of unsatisfactory transaction
  - we can define the Local Trust Value as:

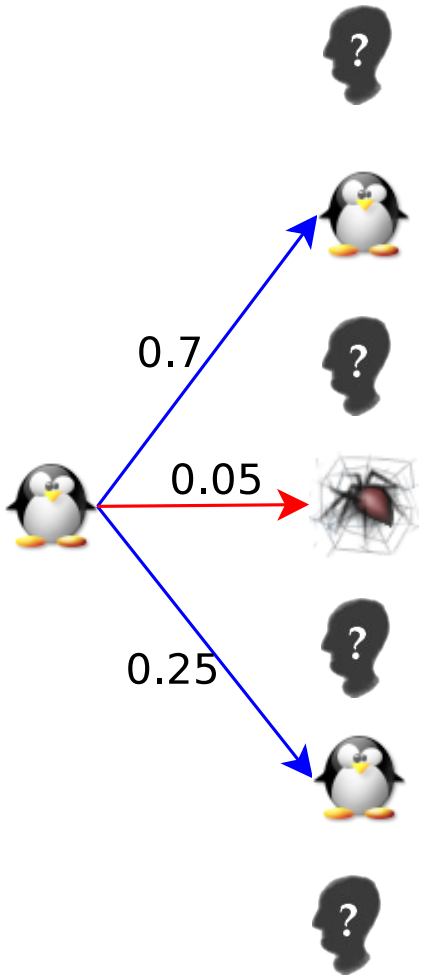
$$s_{ij} = sat(i,j) - unsat(i,j)$$

- Problem: how to aggregate these local scores ?
- normalize these values to obtain probabilities

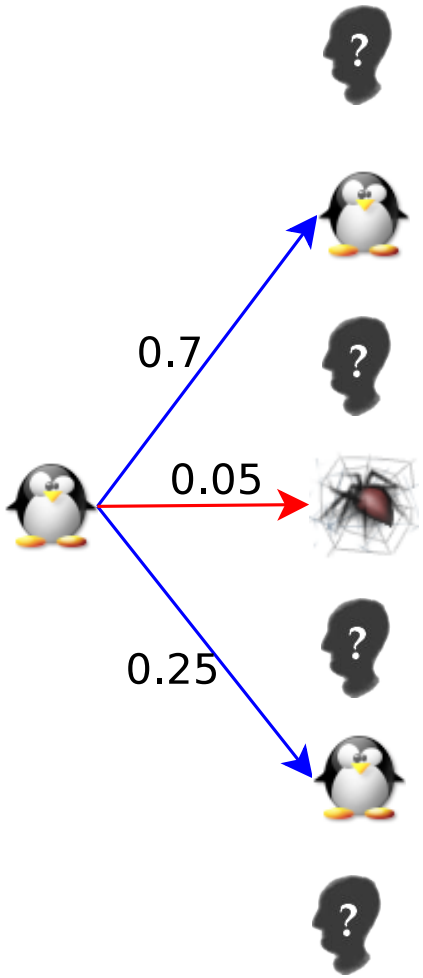
$$0 \quad \leq \quad c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)} \quad \leq \quad 1$$

- Fact: $c_{i1} + c_{i2} + \cdots + c_{in} = 1.$

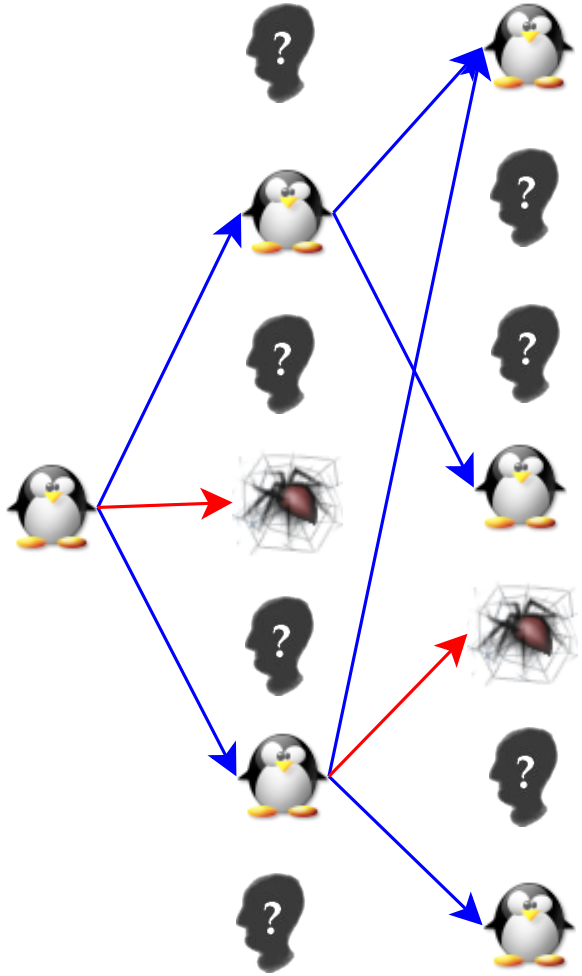
- peer i doesn't know all peers in the network



- peer i doesn't know all peers in the network
- What if peer k is not a neighbor of peer i ?



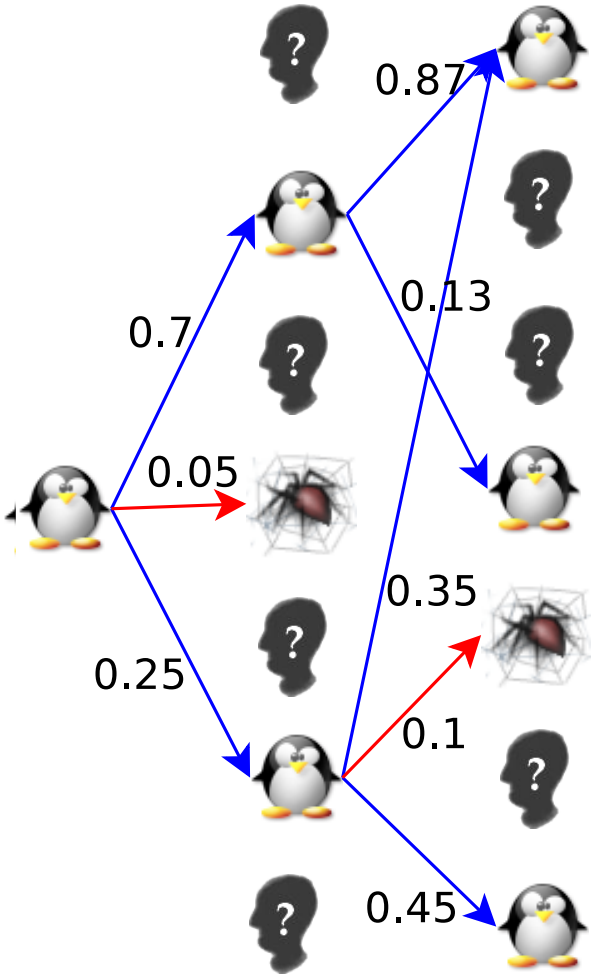
- peer i doesn't know all peers in the network
- What if peer k is not a neighbor of peer i ?
- **Transitive Trust:**
  - ask peers you trust to find trust values of other peers



- peer i doesn't know all peers in the network
- What if peer k is not a neighbor of peer i ?
- **Transitive Trust:**
  - ask peers you trust to find trust values of other peers
  - weight their opinion:

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

  - fact: $0 \leq t_{ik} \leq 1$

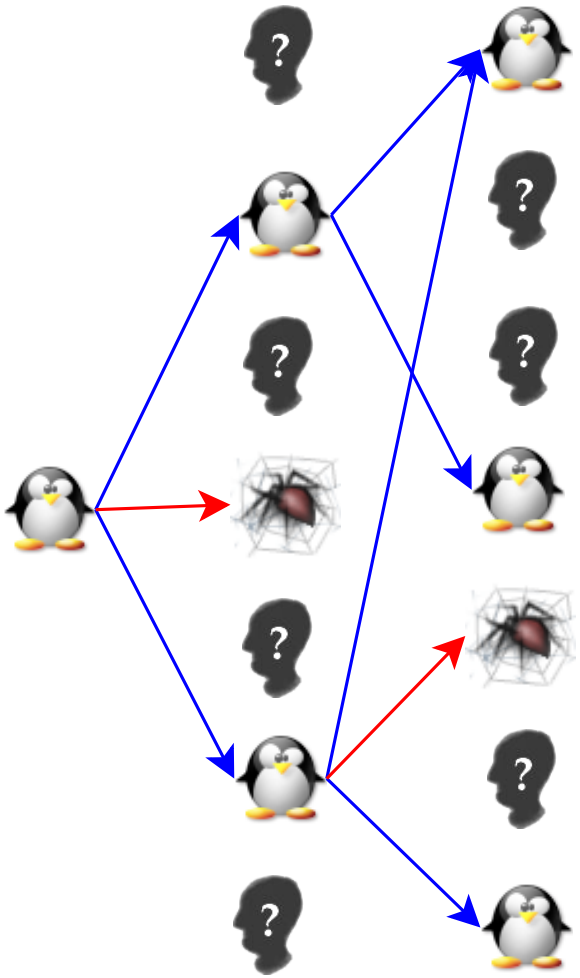


How deep we need to ask ?

How deep we need to ask ?

- ask friends

$$\vec{t_i} = C^T \vec{c_i}$$



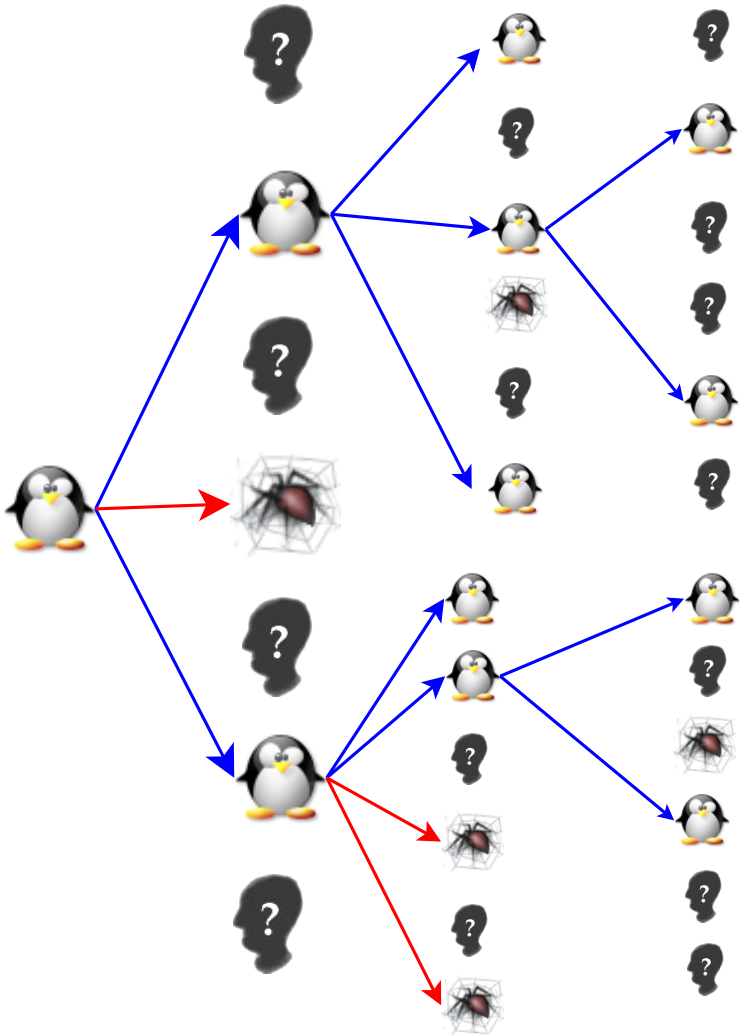
How deep we need to ask ?

- ask friends

$$\vec{t_i} = C^T \vec{c_i}$$

- ask friend's friends

$$\vec{t_i} = (C^T)^2 \vec{c_i}$$



How deep we need to ask ?

- ask friends

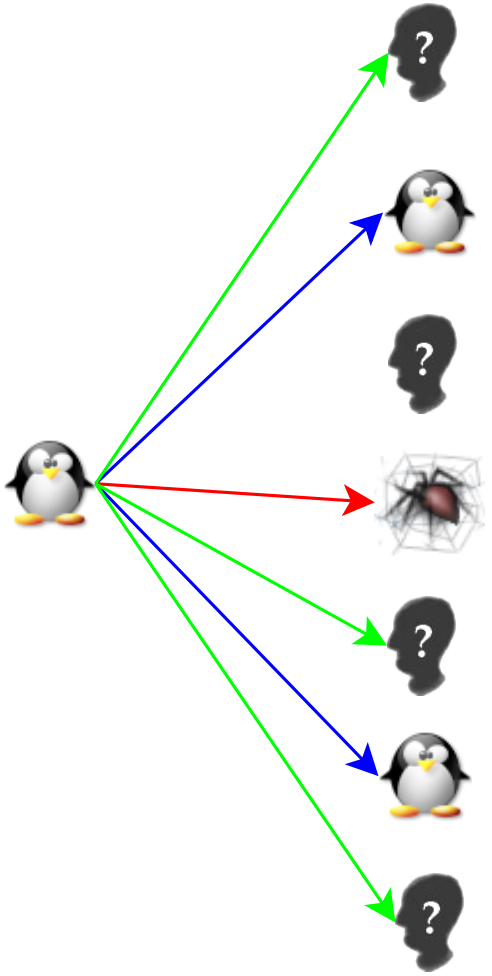
$$\vec{t_i} = C^T \vec{c_i}$$

- ask friend's friends

$$\vec{t_i} = (C^T)^2 \vec{c_i}$$

- keep asking until you're bored

$$\vec{t_i} = (C^T)^n \vec{c_i}$$



- luckily for large values of n all $\vec{t_i}$ are equal: $\vec{t} = \vec{t_i}, \forall i$

- luckily for large values of n all $\vec{t_i}$ are equal: $\vec{t} = \vec{t_i}, \forall i$
- $\vec{t}$ is the Global Trust Vector: it's i^{th} component is i^{th} Global Trust Value

- luckily for large values of n all $\vec{t_i}$ are equal: $\vec{t} = \vec{t_i}, \forall i$
- $\vec{t}$ is the Global Trust Vector: it's i^{th} component is i^{th} Global Trust Value
- similarity with PageRank:
  - Random-Surfer
  - jump from peer i to peer j with probability c_{ij}
  - $\vec{t}$ is the stationary distribution of the MC define by matrix C

- General Overview
- Local and Global Trust Values
- **From Basic to Distributed EigenTrust**
- What about Security ?
- Results and Conclusions

Simple EigenTrust Algorithm

$\vec{t}^{(0)} = \vec{e}$

repeat

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

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

until $\delta < \epsilon$

Simple EigenTrust Algorithm

$\vec{t}^{(0)} = \vec{e}$

repeat

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

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

until $\delta < \epsilon$

Problems ???

Simple EigenTrust Algorithm

$\vec{t}^{(0)} = \vec{e}$

repeat

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

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

until $\delta < \epsilon$

Problems ???

- no prior notion of trust
  - add Pre-trusted peers
  - define a distribution $\vec{p}$ over pre-trusted peers

Simple EigenTrust Algorithm

$\vec{t}^{(0)} = \vec{e}$

repeat

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

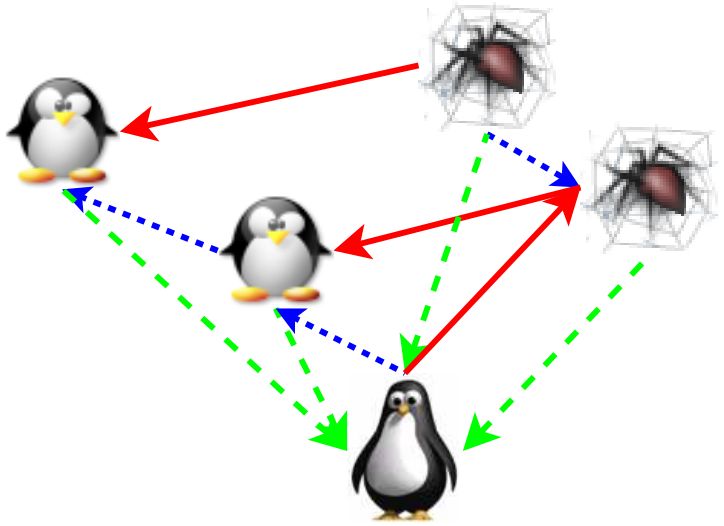
$\delta = \|\vec{t}^{(k+1)} - \vec{t}^{(k)}\|$

until $\delta < \epsilon$

Problems ???

- no prior notion of trust
  - add Pre-trusted peers
  - define a distribution $\vec{p}$ over pre-trusted peers
- inactive peers: peer i is a newcomer or doesn't have any interaction with other peers

$$c_{ij} = \begin{cases} c_{ij}, & \text{if } \sum_j \max(s_{ij}, 0) \neq 0 \\ p_j, & \text{otherwise} \end{cases}$$



- Malicious collectivities: force peers to trust pre-trusted peers

$$\vec{t} = (1 - a)\vec{t} + a\vec{p}$$

- Malicious collectivities: force peers to trust pre-trusted peers

$$\vec{t} = (1 - a)\vec{t} + a\vec{p}$$

looks like PageRank ???

- Malicious collectivities: force peers to trust pre-trusted peers

$$\vec{t} = (1 - a)\vec{t} + a\vec{p}$$

looks like PageRank ???

- Break Collectivities: the Random-Surfer is likely to get stuck when crawling malicious collectivities

- Malicious collectivities: force peers to trust pre-trusted peers

$$\vec{t} = (1 - a)\vec{t} + a\vec{p}$$

looks like PageRank ???

- Break Collectivities: the Random-Surfer is likely to get stuck when crawling malicious collectivities

Basic EigenTrust Algorithm

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

repeat

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

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

until $\delta < \epsilon$

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?
  - put every peer to store its local trust vector $\vec{c_i}$

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?
  - put every peer to store its local trust vector $\vec{c_i}$
  - put every peer to store its own global trust value t_i

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?
  - put every peer to store its local trust vector $\vec{c_i}$
  - put every peer to store its own global trust value t_i
- Idea: write the equation before component-wise:

$$t_i = (1-a)(c_{1i}t_1 + c_{2i}t_2 + \ldots c_{ni}t_n) + ap_i$$

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?
  - put every peer to store its local trust vector $\vec{c_i}$
  - put every peer to store its own global trust value t_i
- Idea: write the equation before component-wise:

$$t_i = (1-a)(c_{1i}t_1 + c_{2i}t_2 + \ldots c_{ni}t_n) + ap_i$$

- Easy to compute: peer i has only few neighbors, so $c_{ji} = 0$ for lots of j

- Can we do it in a distributed manner? Can we store C and $\vec{t}$?
  - put every peer to store its local trust vector $\vec{c_i}$
  - put every peer to store its own global trust value t_i
- Idea: write the equation before component-wise:

$$t_i = (1-a)(c_{1i}t_1 + c_{2i}t_2 + \ldots c_{ni}t_n) + ap_i$$

- Easy to compute: peer i has only few neighbors, so $c_{ji} = 0$ for lots of j
- Key observation: pre-trusted peers will remain anonymous:
  - *nobody need to know their* p_i

Notations:

- A_i: set of peers which have downloaded files form peer i
- B_i: set of peers from which peer i has downloaded files

Distributed EigenTrust Algorithm

foreach peer i

ask peer $j \in A_i$ for c_{ji} and ${t_j}^{(0)} = p_j$

repeat

$t_i^{(k+1)} = (1-a)(c_{1i}t_1^{(k)} + c_{2i}t_2^{(k)} + \ldots c_{ni}t_n^{(k)}) + ap_i$

send your opinion c_{ij} and trust value $t_i^{(k+1)}$ to all peers $j \in B_i$

wait for all peers $j \in A_i$, to respond with their opinion c_{ji} and trust value $t_j^{(k+1)}$

until $|t_i^{(k+1)} - t_i^{(k)}| < \epsilon$

- General Overview
- Local and Global Trust Values
- From Basic to Distributed EigenTrust
- **What about Security ?**
- Results and Conclusions

- Malicious peers: they would lie about their trust value

- Malicious peers: they would lie about their trust value
- Ideas:
  - put a different peer(or better more peers) to compute and store the trust value of a peer: **Score Managers**
  - hide or hash the Score Managers

- Malicious peers: they would lie about their trust value
- Ideas:
  - put a different peer(or better more peers) to compute and store the trust value of a peer: **Score Managers**
  - hide or hash the Score Managers
- How it's working ?
  - peer i want its trust value
  - ask the Score Managers for it
  - if receive different values than vote

- Malicious peers: they would lie about their trust value
- Ideas:
  - put a different peer(or better more peers) to compute and store the trust value of a peer: **Score Managers**
  - hide or hash the Score Managers
- How it's working ?
  - peer i want its trust value
  - ask the Score Managers for it
  - if receive different values than vote
- Why hash them ?
  - every peer compute at least a trust value
  - peers doesn't know how's trust value is computing
  - malicious peers are not encouraged to report wrong values

Notations:

each peer has a number M of SM

$\vec{pos}_i$ the hash mapping of peer i

D_i - the set of peers for whom i is SM

for each $d \in D_i$, c_d^i is LTV of d maintained by i

Secure EigenTrust Algorithm

foreach peer i **do**

submit local trust values to score managers

collect local trust values

submit local trust values c_{dj} to score managers

foreach daughter peer $d \in D_i$ **do**

ask peer $j \in A_d^i$ for $c_{jd}p_j$

repeat

$t_d^{(k+1)} = (1-a)(c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \ldots c_{nd}t_n^{(k)}) + ap_d$

send your opinion c_{dj} and trust value $t_d^{(k+1)}$ to all peers $j \in B_d^i$

wait for all peers $j \in A_i^d$, to respond with their opinion c_{jd} and trust value $t_j^{(k+1)}$

until $|t_d^{(k+1)} - t_d^{(k)}| < \epsilon$

- General Overview
- Local and Global Trust Values
- From Basic to Distributed EigenTrust
- What about Security ?
- Results and Conclusions

- When a peer get responses from multiple peers: $\{t_1, t_2, \ldots, t_R\}$
  - **deterministic:** Choose the one with highest trust value: $t_{\text{max}} = \max_k (t_k)$
  - **probabilistic:** Choose a peer with probability $\frac{t_i}{\sum_k t_k}$; if peer i has $t_j = 0$ choose her with probability 0.1

- When a peer get responses from multiple peers: $\{t_1, t_2, \ldots, t_R\}$
  - **deterministic:** Choose the one with highest trust value: $t_{\text{max}} = \max_k (t_k)$
  - **probabilistic:** Choose a peer with probability $\frac{t_i}{\sum_k t_k}$; if peer i has $t_j = 0$ choose her with probability 0.1

Load Distribution Algorithm

get $T = \{t_1, t_2, \ldots, t_R\}$

repeat

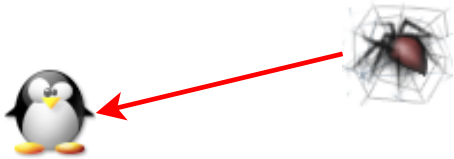
choose a peer j with $t_j \in T$

if receive inauthentic file form j **then**

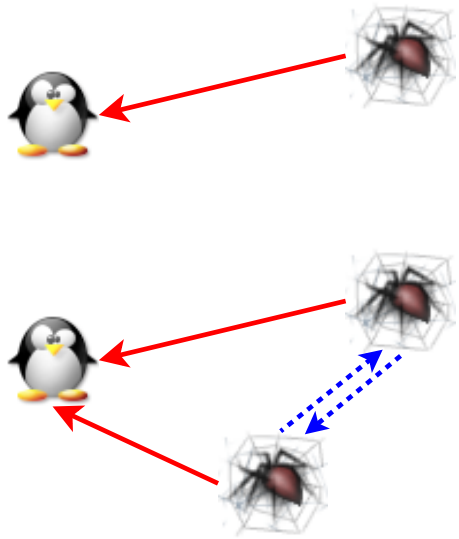
delete t_j from T

until authentic file received

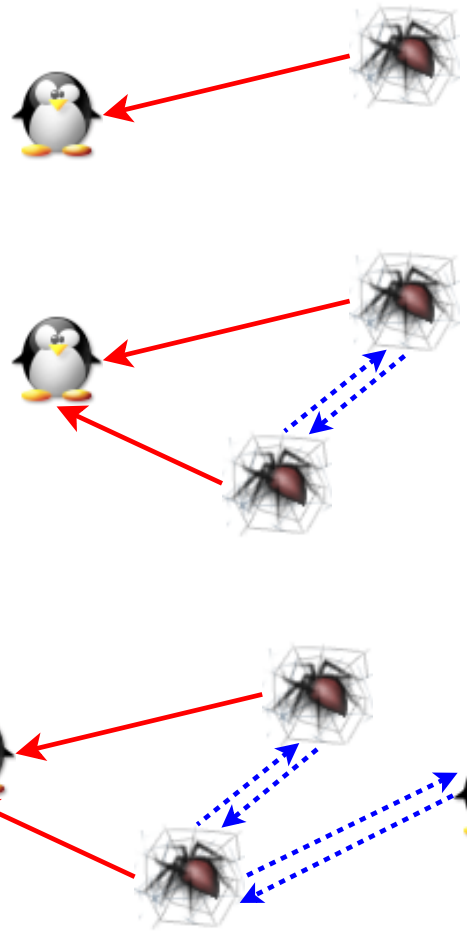
- **Malicious Individuals:**
  - peers that always provide inauthentic files.



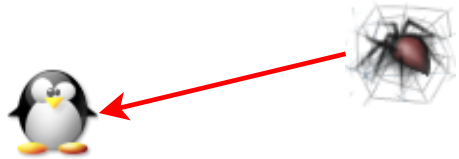
- **Malicious Individuals:**
  - peers that always provide inauthentic files.
- **Malicious Collectives:**
  - peers that always provide inauthentic files.
  - know each other.
  - give each other good opinions, and give other peers bad feedback.

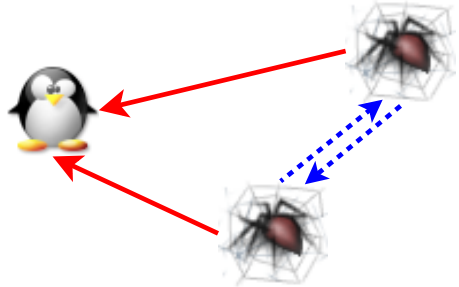


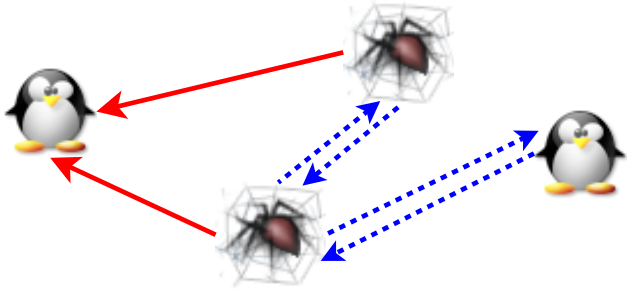
- **Malicious Individuals:**
  - peers that always provide inauthentic files.
- **Malicious Collectives:**
  - peers that always provide inauthentic files.
  - know each other.
  - give each other good opinions, and give other peers bad feedback.
- **Camouflaged Collectives:**
  - peers provide authentic files some of the time to trick good peers into for achieving good feedback.

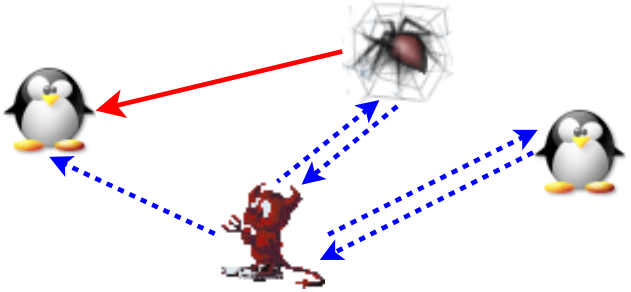


- **Malicious Individuals:**
  - peers that always provide inauthentic files.
- **Malicious Collectives:**
  - peers that always provide inauthentic files.
  - know each other.
  - give each other good opinions, and give other peers bad feedback.
- **Camouflaged Collectives:**
  - peers provide authentic files some of the time to trick good peers into for achieving good feedback.
- **Malicious Spies:**
  - some members of the collective give good files all the time, but give good feedback to malicious peers.









- **highs:**
  - Reduces number of **inauthentic files** on the network.
  - Low overhead.
  - Try to minimize the number of malicious peers: **robustness**

- **highs:**
  - Reduces number of **inauthentic files** on the network.
  - Low overhead.
  - Try to minimize the number of malicious peers: **robustness**
- **lows:**
  - what happens with a dynamically changing network ???