

Incentivi alla cooperazione

Sistemi di reputazione

Matteo Dell'Amico
dellamico@disi.unige.it



Sistemi Distribuiti – P2P
A.A. 2006-07
27 novembre 2006

Indice

- 1 Feldman et al.
 - Introduzione
 - Esperimento
 - Risultati
- 2 EigenTrust
 - Idea
 - Algoritmo
 - Esperimenti
- 3 Conclusioni

Feldman et al.

M. Feldman, K. Lai, I. Stoica e J. Chuang, 2004

- Simulazione della funzione di reputazione **MaxFlow** in una rete P2P.

| | | |
|-----|------|-----|
| C\S | C | D |
| C | 7,-1 | 0,0 |

- Gioco analogo a **donatore/ricevente**, visto precedentemente.
- **Defezioni non tracciabili**: chi rifiuta di collaborare non viene riconosciuto.
- **Popolazione dinamica**: i peer entrano nella rete, ne escono, cambiano strategia.

Dinamiche temporali

Round

- In ogni round ogni peer:
 - 1 agisce come **client**;
 - 2 agisce come **server**;
 - 3 **sceglie la strategia** per il prossimo round.

Nuova strategia

- Con una certa probabilità, il peer sceglie una di queste possibilità:
 - muta, scegliendo una nuova strategia;
 - impara, adottando la migliore strategia;
 - lascia il sistema, e viene sostituito da un nuovo arrivato con la stessa strategia.

Dinamiche temporali

Round

- In ogni round ogni peer:
 - ① agisce come **client**;
 - ② agisce come **server**;
 - ③ **sceglie la strategia** per il prossimo round.

Nuova strategia

- Con una certa probabilità, il peer sceglie una di queste possibilità:
 - muta, scegliendo una nuova strategia;
 - impara, adottando la migliore strategia;
 - lascia il sistema, e viene sostituito da un nuovo arrivato con la stessa strategia.

Strategie

Defezionatore

Cooperatore

Storia privata Valuta la generosità di un nodo i come il **rapporto** tra servizio che gli ha fornito e ha ricevuto $g(i) = p_i/c_i$, e la normalizza rispetto alla sua stessa generosità. j coopera con i con probabilità

$$g_j(i) = \min \left(\frac{g(i)}{g(j)}, 1 \right)$$

Storia condivisa Funziona analogamente alla storia privata, valutando però **l'intera storia passata** del nodo.

Strategie

Defezionatore

Cooperatore

Storia privata Valuta la generosità di un nodo i come il **rapporto** tra servizio che gli ha fornito e ha ricevuto $g(i) = p_i/c_i$, e la normalizza rispetto alla sua stessa generosità. j coopera con i con probabilità

$$g_j(i) = \min \left(\frac{g(i)}{g(j)}, 1 \right)$$

Storia condivisa Funziona analogamente alla storia privata, valutando però l'intera storia passata del nodo.

Strategie

Defezionatore

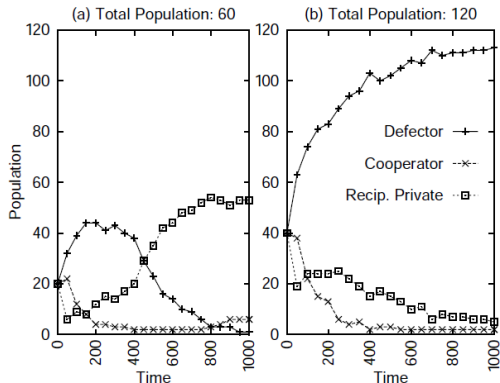
Cooperatore

Storia privata Valuta la generosità di un nodo i come il **rapporto** tra servizio che gli ha fornito e ha ricevuto $g(i) = p_i/c_i$, e la normalizza rispetto alla sua stessa generosità. j coopera con i con probabilità

$$g_j(i) = \min \left(\frac{g(i)}{g(j)}, 1 \right)$$

Storia condivisa Funziona analogamente alla storia privata, valutando però **l'intera storia passata** del nodo.

Storia privata



Parametri

- Probabilità di mutazione 0, apprendimento 0.05, turnover 0.0001.

Collusione

- Un **Sybil attack** può sovvertire il risultato calcolato dalla storia condivisa. Maxflow, invece, è sybilproof.
- Il peso degli archi nella WoT è il **numero di transazioni** andate a buon fine.

Collusione

- Un **Sybil attack** può sovvertire il risultato calcolato dalla storia condivisa. Maxflow, invece, è sybilproof.
- Il peso degli archi nella WoT è il **numero di transazioni** andate a buon fine.

Strategia soggettiva

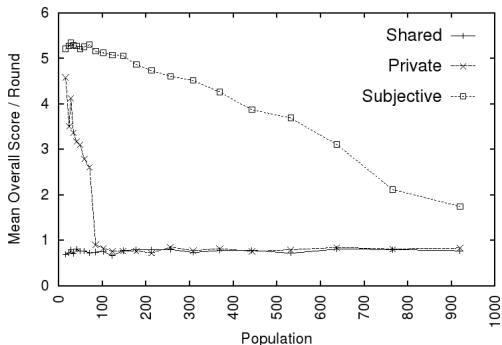
Strategia soggettiva

- Valuta la generosità di un nodo come il **flusso** che lo collega ad esso, e la rapporta al flusso nella direzione inversa. Quindi, il nodo j coopera con i con probabilità

$$g_j(i) = \min \left(\frac{\text{MaxFlow}(j, i)}{\text{MaxFlow}(i, j)}, 1 \right).$$

- MaxFlow ha **complessità** in $O(V^3)$. Per evitare un sovraccarico, se la durata media del calcolo ha superato un limite fissato, viene restituito come risultato un flusso nullo.

Risultati sperimentali



- Il punteggio medio scende al crescere della popolazione a causa del vincolo sul tempo di esecuzione di MaxFlow (in questo caso, la visita di 100 nodi).

EigenTrust

S. D. Kamvar, M. T. Schlosser ed H. Garcia-Molina, 2003

- Sistema di reputazione ispirato a **PageRank**.
- Modello: *random walk* sulla web of trust.
- I nodi più importanti hanno **più probabilità** di venire visitati.
- Gli archi provenienti dai nodi più importanti sono **più importanti**.
- Per ottenere una reputazione alta, **conviene fornire un buon servizio** ai nodi “buoni”.

L'idea

- La WoT viene **normalizzata** in maniera tale da avere, per ogni nodo i , una somma di valori sugli archi uscenti c_{ij} pari ad 1.
- I punti di partenza dell'algoritmo sono fissati: una certa quantità k di nodi "**pre-trusted**".
- Il cammino casuale **si ferma** con probabilità α , e prosegue con probabilità $1 - \alpha$.
- I link più "pesanti" hanno **più probabilità di essere seguiti**.
- Siamo interessati a calcolare la probabilità che **il cammino si fermi** in ogni nodo.
- Per il momento, supponiamo che ogni nodo calcoli la sua stessa reputazione.

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - ① Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t_j^{(k-1)}$.
 - ② Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - ③ Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - ① Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t^{(k-1)}$.
 - ② Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - ③ Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - ① Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t_j^{(k-1)}$.
 - ② Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - ③ Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - 1 Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t^{(k-1)}$.
 - 2 Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - 3 Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - 1 Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t^{(k-1)}$.
 - 2 Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - 3 Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - 1 Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t^{(k-1)}$.
 - 2 Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - 3 Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

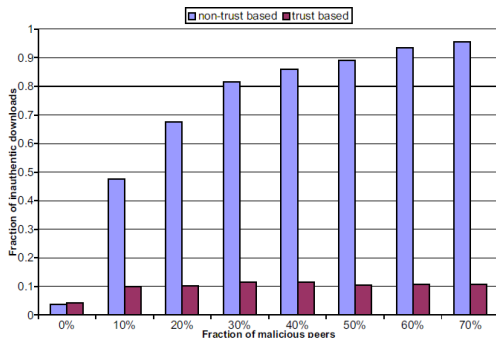
L'algoritmo

- Ogni nodo conosce il suo valore di partenza p_i : 0 se non è un nodo pre-trusted, $1/k$ altrimenti.
- Si inizializza il valore $t_i^{(0)}$ a p_i .
- Finché l'algoritmo non converge, il nodo i al ciclo k esegue:
 - ① Si contattano tutti i nodi con archi entranti in i e si chiede il loro valore di $t^{(k-1)}$.
 - ② Si calcola $t_i^{(k)} = (1 - \alpha) \sum_j (t_j^{(k-1)} c_{ij}) + \alpha p_i$.
 - ③ Si invia $t_i^{(k)}$ a tutti i nodi verso cui escono archi da i .
- Il valore finale di t_i è la reputazione associata ad i .

EigenTrust sicuro

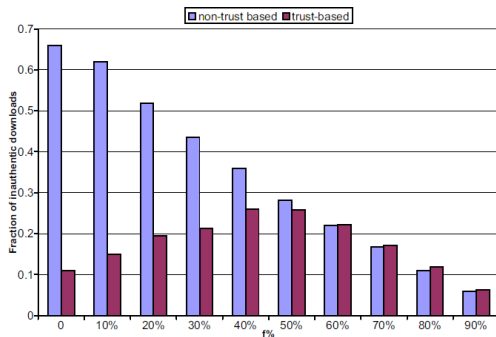
- È naturale che i nodi possano **mentire**, nel momento in cui devono valutare la loro stessa reputazione.
- Se si ha a disposizione una DHT, si può fare calcolare la reputazione del nodo con ID i ad $h_1(i), h_2(i), \dots, h_n(i)$, con $h_1 \dots h_n$ funzioni hash distinte.
- La reputazione di i viene ottenuta contattando i nodi responsabili ed effettuando un **voto a maggioranza**.
- Come in tutti i casi di questo tipo, il sistema è suscettibile al *Sybil attack*. È importante l'**assegnazione degli ID**.

Nodi maligni



- Simulazione **rete di filesharing** stile Gnutella.
- Nodi maligni **collusi**, che riportano valori falsi di fiducia e effettuano upload di file non autentici.

Nodi maligni (2)



- Nodi maligni collusi che effettuano upload di file autentici con **probabilità variabile**.

Conclusioni

- I sistemi di reputazione sono sicuramente **promettenti** per progettare sistemi P2P su larga scala.
- Allo stato dell'arte, nessun sistema è perfetto:
 - I sistemi semplici basati sulla reciprocazione diretta non sono ottimali e funzionano solo in **casi ristretti** (per esempio, lo scambio di file di grandi dimensioni);
 - MaxFlow è un algoritmo **costoso**, e la sua efficacia decresce con la dimensione della rete su cui viene calcolata;
 - EigenTrust è in qualche maniera suscettibile al Sybil attack, e richiede dei nodi **“pre-trusted”** che sono problematici da gestire.
- Inoltre, nessun sistema di reputazione non banale è stato implementato in reti di grandi dimensioni. Vari problemi “pratici” (p. es., scheduling) sono stati al momento ignorati.