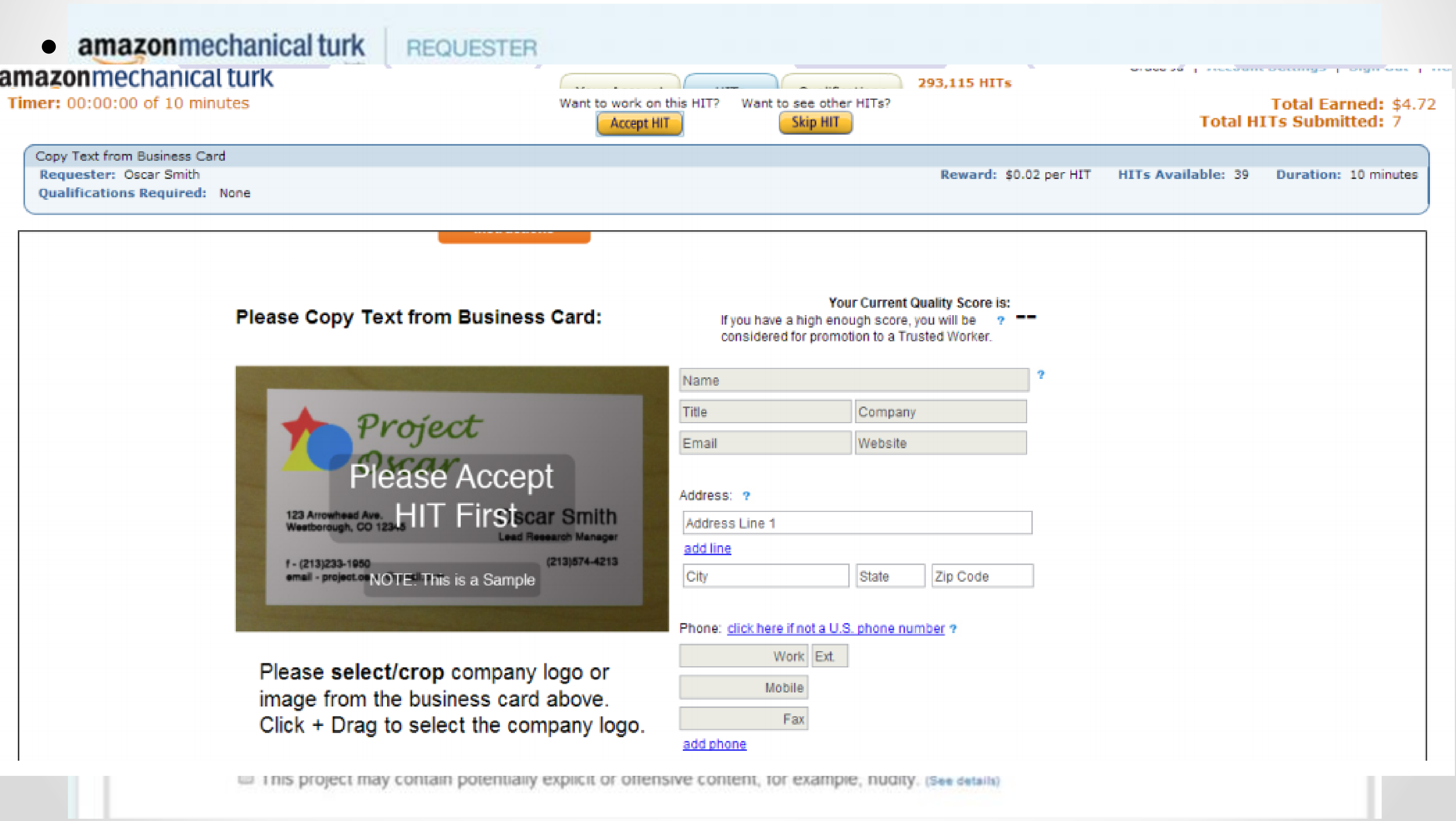
Reliability Enhanced Social Crowdsourcing

Wei Chang and Jie Wu
Temple University
Email: wei.chang@temple.edu

Introduction

- Crowdsourcing:
  - Job owner partitions a tedious work into pieces, and outsources them onto a crowdsourcing platform.
  - Independent freelances search and take up some subworks. After finishing sub-works, they return results to the platform.
  - Centralized platform: Amazon Mturk

Introduction

- 

amazonmechanical turk | REQUESTER

amazonmechanical turk

Timer: 00:00:00 of 10 minutes

293,115 HITs

Want to work on this HIT? Accept HIT

Want to see other HITs? Skip HIT

Total Earned: $4.72
Total HITs Submitted: 7

Copy Text from Business Card
Requester: Oscar Smith
Qualifications Required: None
Reward: $0.02 per HIT
HITs Available: 39
Duration: 10 minutes

Please Copy Text from Business Card:

Your Current Quality Score is: --
If you have a high enough score, you will be considered for promotion to a Trusted Worker.

Name
Title | Company
Email | Website

Address:
Address Line 1
add line
City | State | Zip Code

Phone: click here if not a U.S. phone number
Work | Ext.
Mobile
Fax
add phone

Please **select/crop** company logo or image from the business card above. Click + Drag to select the company logo.

This project may contain potentially explicit or offensive content, for example, nudity. (See details)

Introduction

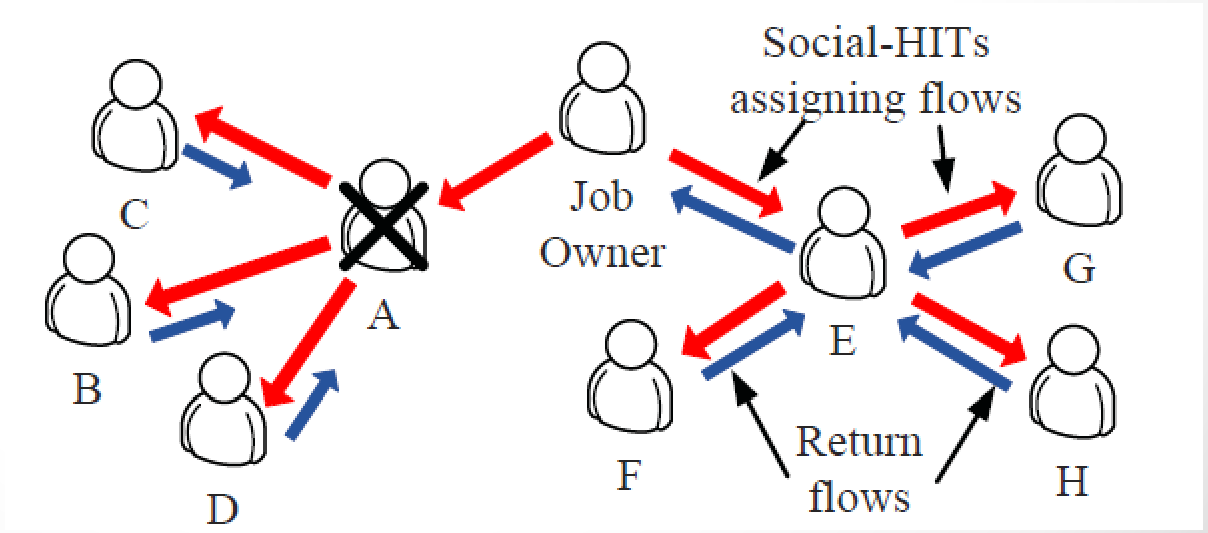
- Crowdsourcing
- Problem with the conventional crowdsourcing:
  - Although crowdsourcing brings more knowledge diversity and a large amount of labor force, the independent feature of workers causes the problem that it can only process simple and independent works. For complex tasks, we need trusted experts.
  - It is hard for a newly created task to attract enough participants in a relatively short time, unless the task owner gives a very attractive payment. We need priority for our tasks.

Introduction

- Crowdsourcing
- Problem with the conventional crowdsourcing:
  - trusted experts
  - prioritized tasks
- Social Crowdsourcing (SC): explores the social relations among participants
  - Add a new dimension, sociality, to existing platforms.
  - A job can be completed, via iterative recruitment of workers through social ties.
  - Unlike the existing systems, workers of SC are not independent.

Introduction

- Crowdsourcing
- Problem with the conventional crowdsourcing
- Social Crowdsourcing (SC)
- Reliability issue with SC
    - Early return
    - Offline
    - Drop out



Introduction

- Crowdsourcing
- Problem with the conventional crowdsourcing
- Social Crowdsourcing (SC)
- Reliability issue with SC
- Reliability enhanced SC
    - Preplanned redundant return paths in SC
    - Returning rules

System model

- Social Crowdsourcing models the job's outsourcing procedure via the process of iteratively recruiting friends' friends.
- Job owner: creates social-HIT (i.e. task)
- Human Worker:
  - Locally processes the social-HIT
  - Further propagates the social-HIT to others, and collects results
  - Return the results to the participant, who gave the social-HIT

System model

- Social Crowdsourcing models the job’s outsourcing procedure via the process of iteratively recruiting friends’ friends.
- Job owner: creates social-HIT (i.e. task)
- Human Worker:
  - Locally processes the social-HIT
  - Further propagates the social-HIT to others, and collects results
  - Return the results to the participant, who gave the social-HIT
- Worker status:
  - Awake, sleep, done, and dead

Social Crowdsourcing: design details

- A social-HIT: $\langle JobId, Father, LifeTime, Hop, Instruct \rangle$
  - **JobID**: unique id of the original job
  - **Father**: the participant who gave the social-HIT
  - **LifeTime**: timely clean-up the starved job
  - **Hop**: the number of remaining hops
  - **Instruct**: job description and **specific returning conditions**

Social Crowdsourcing: design details

- A social-HIT: $\langle JobId, Father, LifeTime, Hop, Instruct\rangle$
- Returning conditions:

  For relay nodes (Hop is non-zero):

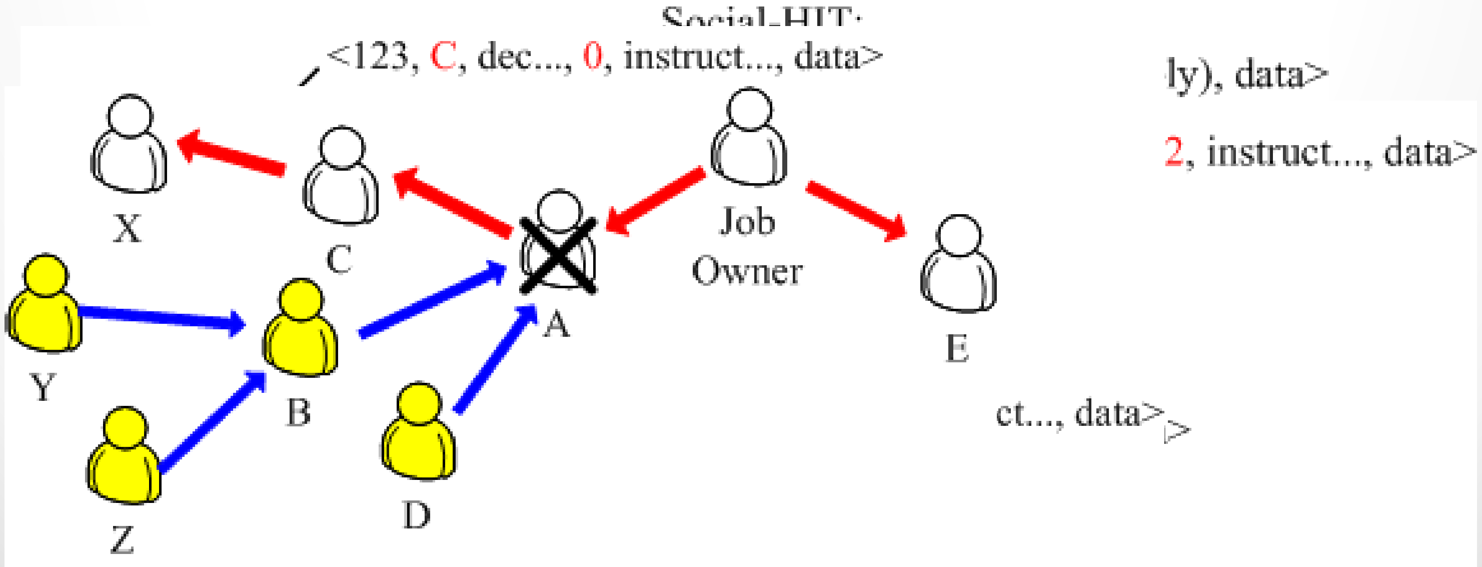
  - When a node is awake and receives **c** replies, the node immediately returns his result;
  - If the node is in sleep and receives more than c replies during sleeping, it should return all of them when it wakes up.

  For the non-relay nodes (Hop equals zero):

  - When a node is awake and finish the work, it should immediately return the result.

Social Crowdsourcing: design details

- A social-HIT: $\langle JobId, Father, LifeTime, Hop, Instruct \rangle$
- Returning conditions for relay nodes:
  - *When a node is awake and receives **c** replies, the node immediately returns his result;*
  - *If the node is in sleep and receives more than c replies during sleeping, it should return all of them when it wakes up.*



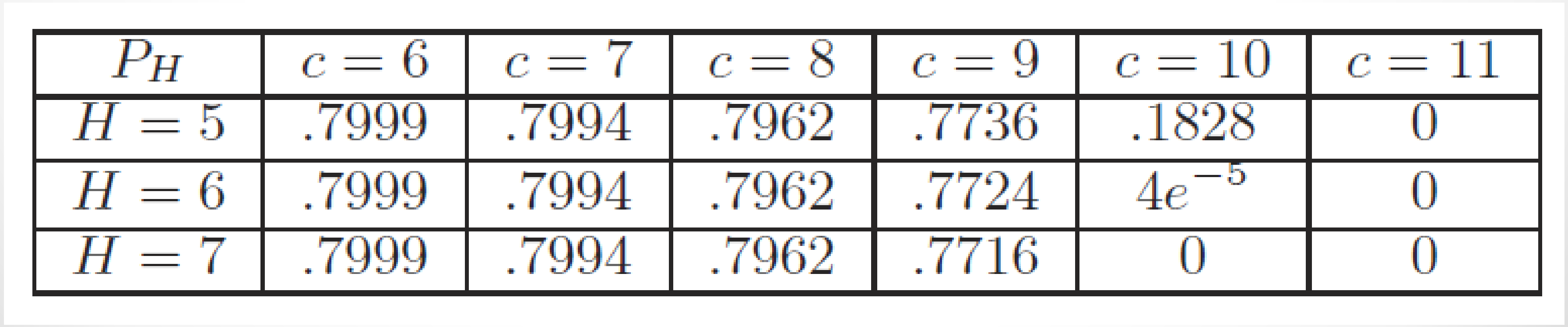
Reliability issue

- Successful return rate:
  - P_i : the probability that a node with Hop=i successfully returns its subtree's results to its father node.
  - r: average number of child
  - R: reliability

$$P_i = R \cdot \sum_{j=c}^{r} \binom{r}{j} \cdot P_{i-1}^{j} \cdot (1 - P_{i-1})^{r-j}$$

Reliability issue

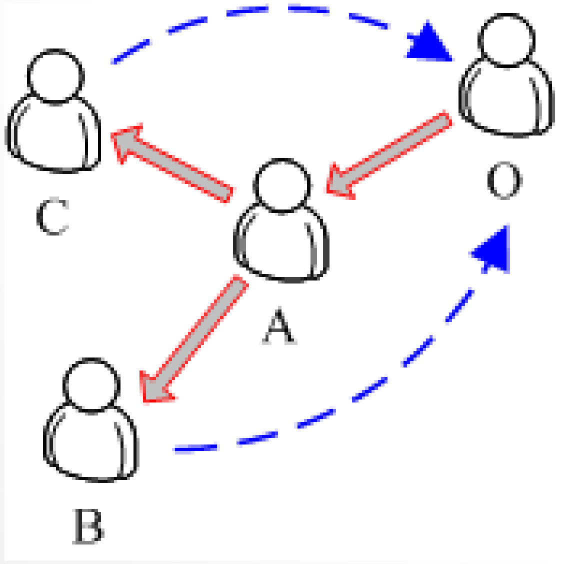
- Successful return rate:
  - P_i : the probability that a node with Hop=i successfully returns its subtree's results to its father node.
  - r: average number of child
  - R: reliability
- For example, when r=15 and R=0.8, we have:



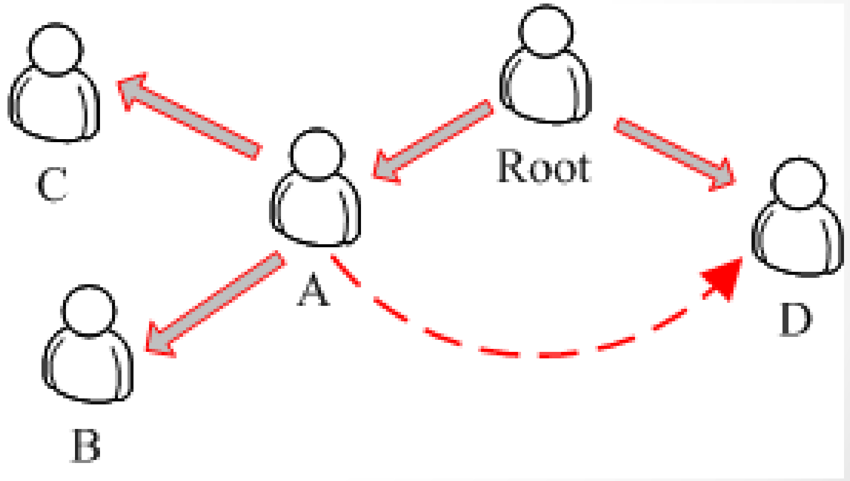
P_H	$c = 6$	$c = 7$	$c = 8$	$c = 9$	$c = 10$	$c = 11$
$H = 5$	.7999	.7994	.7962	.7736	.1828	0
$H = 6$	.7999	.7994	.7962	.7724	$4e^{-5}$	0
$H = 7$	.7999	.7994	.7962	.7716	0	0

Reliability enhanced SC: GFC structure

- Grandpa, Father, Current node structure (GFC) represent a triangle relation in which a non-root node records the identities of its father (a primary return node) and grandfather/sibling (a backup return node).



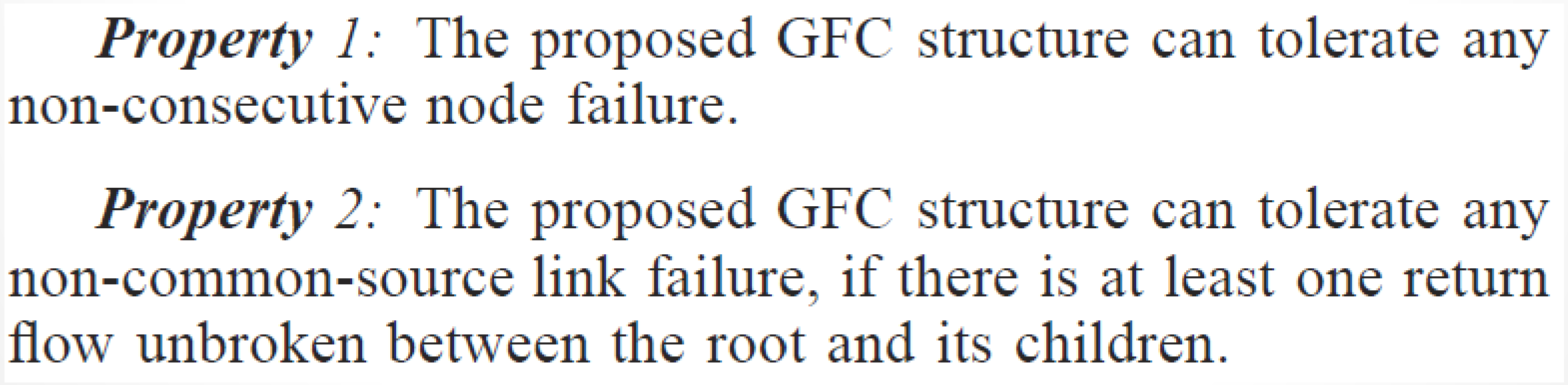
Children of non-root node



Children of the root node

Reliability enhanced SC: GFC structure

- Grandpa, Father, Current node structure (GFC) represent a triangle relation in which a non-root node records the identities of its father (a primary return node) and grandfather/sibling (a backup return node).

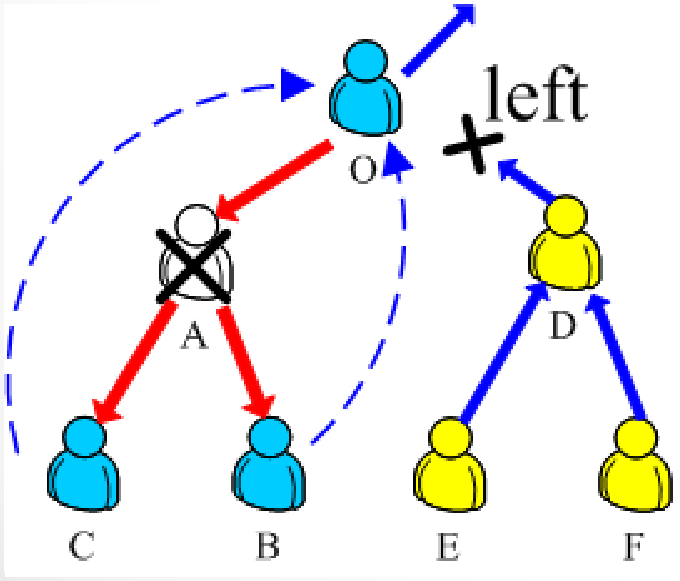


Property 1: The proposed GFC structure can tolerate any non-consecutive node failure.

Property 2: The proposed GFC structure can tolerate any non-common-source link failure, if there is at least one return flow unbroken between the root and its children.

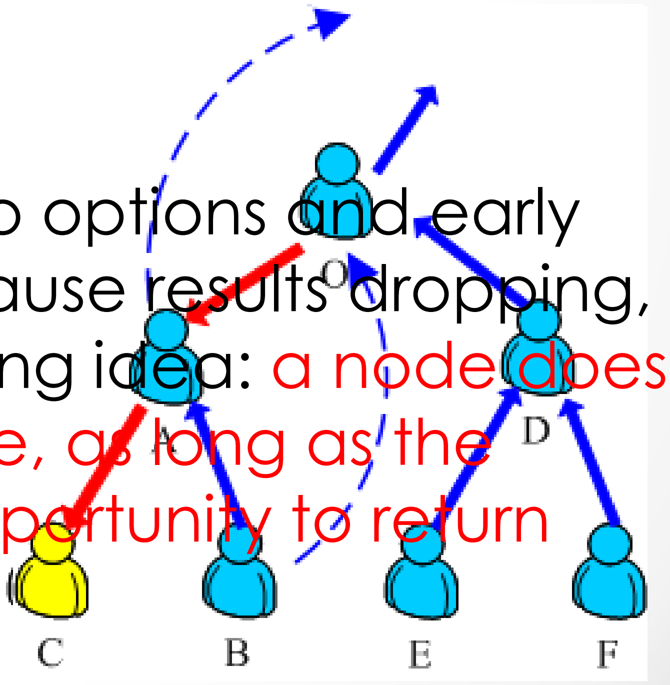
When should we use the backup paths?

- Using the backup path too early may cause many potential results dropped off.



When should we use the backup paths?

- Using the backup path too early may cause many potential results dropped off.
- Returning results too late may result in both father and grandpa nodes left.

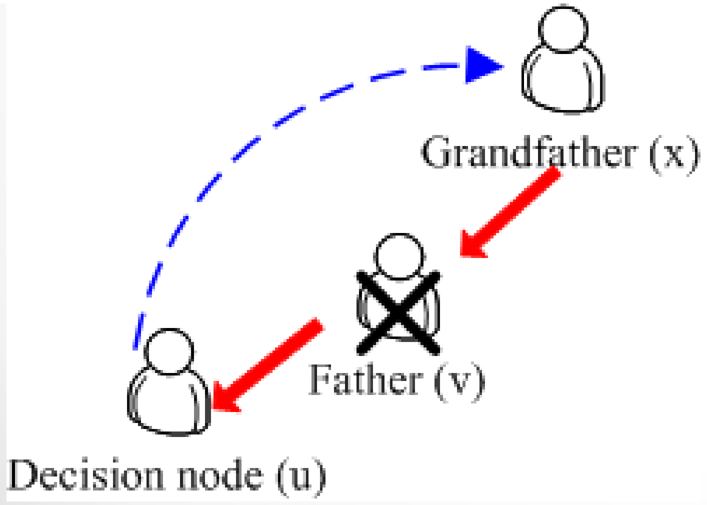


Since each node has only two options and early using the backup one may cause results dropping, GFC-SNCA adopts the following idea: a node does not use its backup return node, as long as the primary one have enough opportunity to return data to a higher-level node.

GFC structure-based returning rules

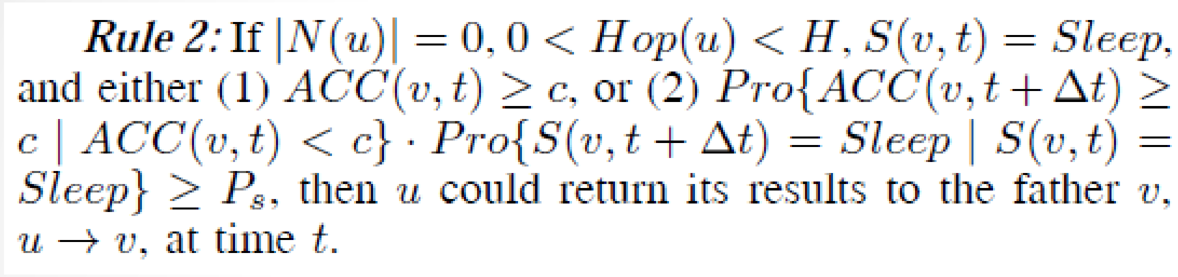
Rule 1: For node u satisfied $S(u,t) = Awake$, $ACC(u,t) \geq c$, and either (1) $S(v,t) = Done/Dead$ or (2) $ACC(x,t) > C_1$, $ACC(v,t) < C_2$, if its grandfather node x satisfies one of the following three conditions, then u should give the results to x instead of v.

1) If $S(x,t) = Sleep$, $ACC(x,t) \geq c$, then $u \to x$ at t.
2) If $Pro\{ACC(x,t+\Delta t) \geq c \mid ACC(x,t) < c\} \cdot Pro\{S(x,t+\Delta t) = Sleep \mid S(x,t) = Sleep\} \geq P_s$, then $u \to x$ at time t.
3) If the number of collected results of x does not change in the next k consecutive time intervals, and $ACC(x,t+k\cdot\Delta t) > C_1$, then $u \to x$ at $t+k\cdot\Delta t$.

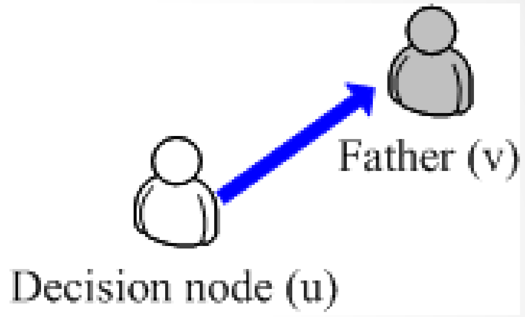


- Current decision node:
  - Wake + have collected enough results
- Father node:
  - Dead/ done
  - The number of collected results are significantly less then grandfather node
- Grandfather node:
  - Sleeping + have collected enough results
  - Very likely to collected enough results and wake up before next checking time
  - Have collected a large portion from its children + the number of collected results does not changed in a period of time

GFC structure-based returning rules

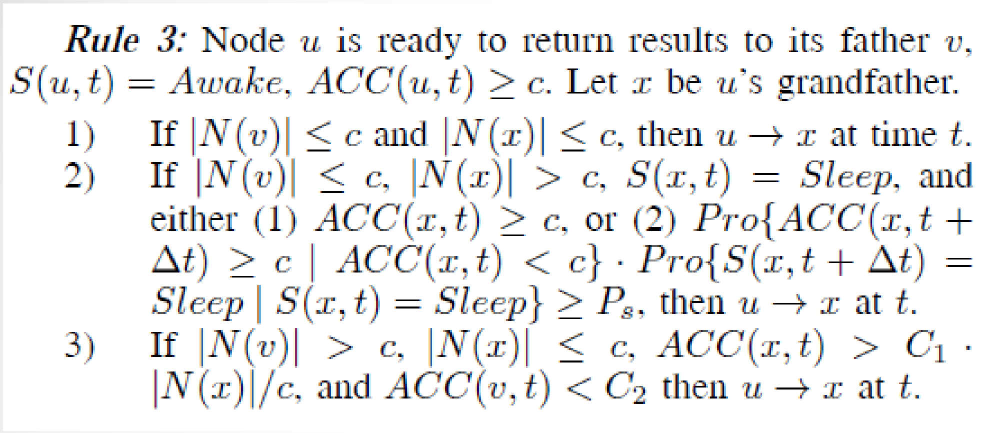


Rule 2: If $|N(u)| = 0, 0 < Hop(u) < H, S(v,t) = Sleep$, and either (1) $ACC(v,t) \geq c$, or (2) $Pro\{ACC(v,t+\Delta t) \geq c \mid ACC(v,t) < c\} \cdot Pro\{S(v,t+\Delta t) = Sleep \mid S(v,t) = Sleep\} \geq P_s$, then u could return its results to the father v, $u \rightarrow v$, at time t.



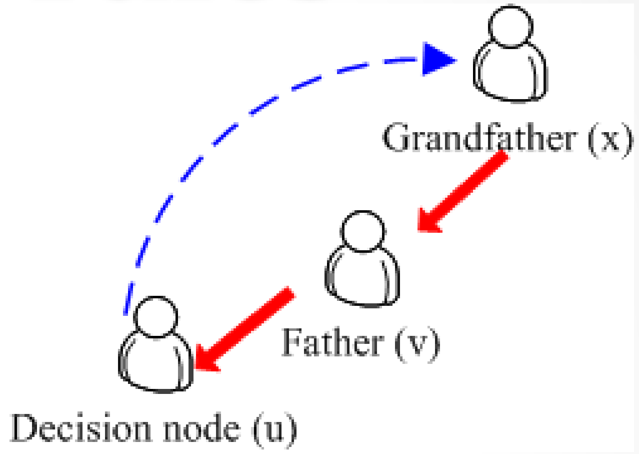
- Current decision node:
  - No child and Hop value in the social-hit is non-zero
- Father node:
  - Sleeping + have collected enough number of results
  - It is very likely for the father node to collected enough results and wake up before next checking time

GFC structure-based returning rules



Rule 3: Node u is ready to return results to its father v, $S(u,t) = Awake$, $ACC(u,t) \geq c$. Let x be u's grandfather.

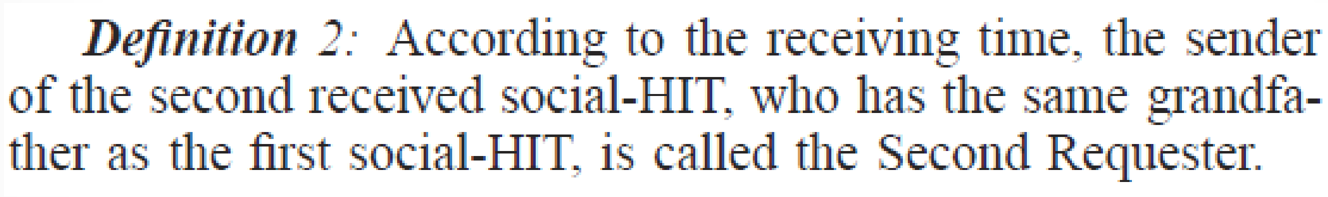
1) If $|N(v)| \leq c$ and $|N(x)| \leq c$, then $u \to x$ at time t.
2) If $|N(v)| \leq c$, $|N(x)| > c$, $S(x,t) = Sleep$, and either (1) $ACC(x,t) \geq c$, or (2) $Pro\{ACC(x,t+\Delta t) \geq c \mid ACC(x,t) < c\} \cdot Pro\{S(x,t+\Delta t) = Sleep \mid S(x,t) = Sleep\} \geq P_s$, then $u \to x$ at t.
3) If $|N(v)| > c$, $|N(x)| \leq c$, $ACC(x,t) > C_1 \cdot |N(x)|/c$, and $ACC(v,t) < C_2$ then $u \to x$ at t.



- Current node will directly submit its result to the grandfather node if one of the following cases is satisfied:
- Case 1: both father and grandfather do not have enough children
- Case 2: father will never collect enough result; the grandfather node is sleeping, and has or will have collect enough results before next checking time
- Case 3: collecting progress of father node is too slow while grandfather node has collected a majority of the returns from its children

Extension: Second Requester-based backup Path

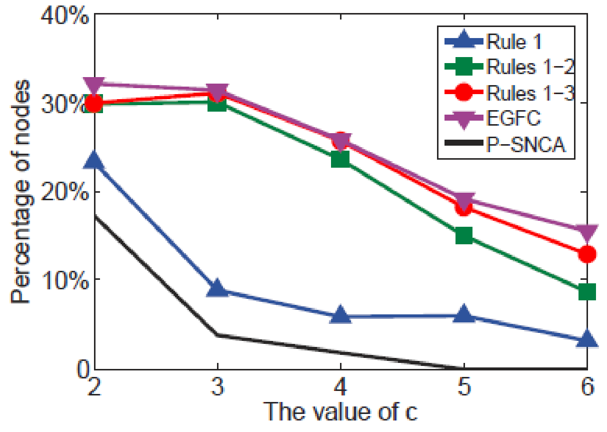
- Consider that, if v fails, plenty of results from v's children will flock to the v's father, x, which may cause x's return slots being quickly used up.



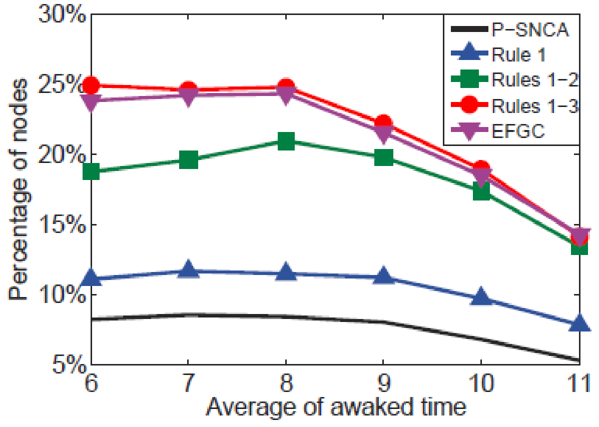
Definition 2: According to the receiving time, the sender of the second received social-HIT, who has the same grandfather as the first social-HIT, is called the Second Requester.

- Returning rules for using the second requester-based path are the same as the ones for grandfather nodes (GFC structure).

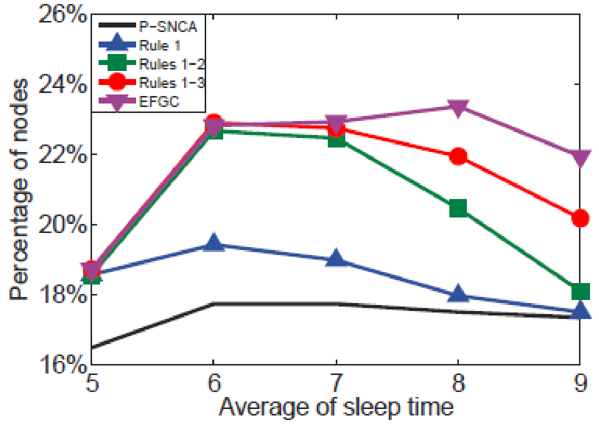
Evaluation



(a) Return requirement c

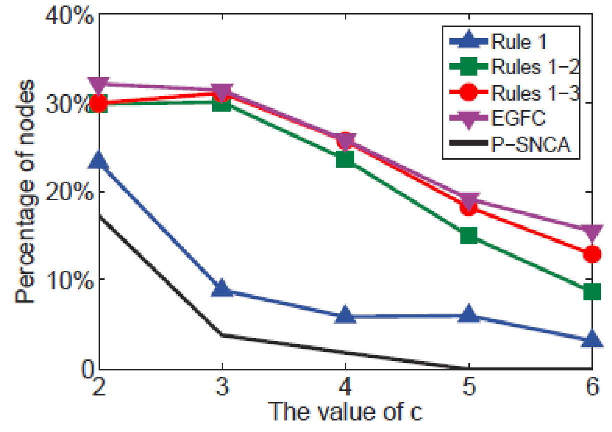


(b) Ave. length of awake time

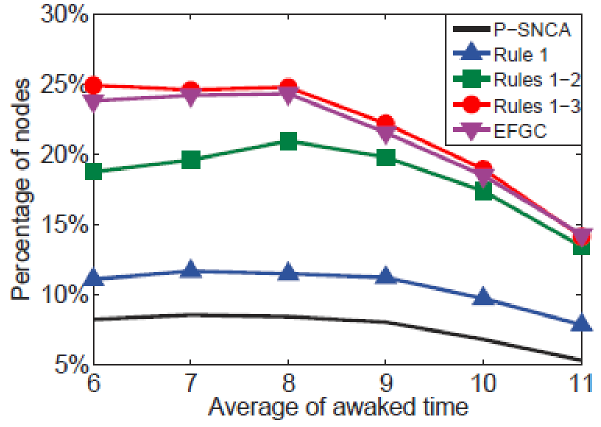


(c) Ave. length of sleep time

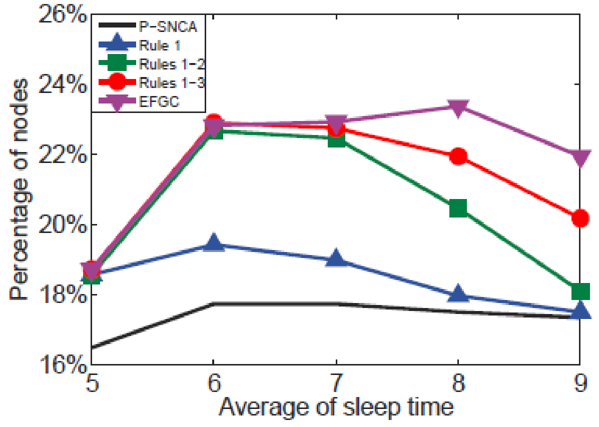
Evaluation



(a) Return requirement c

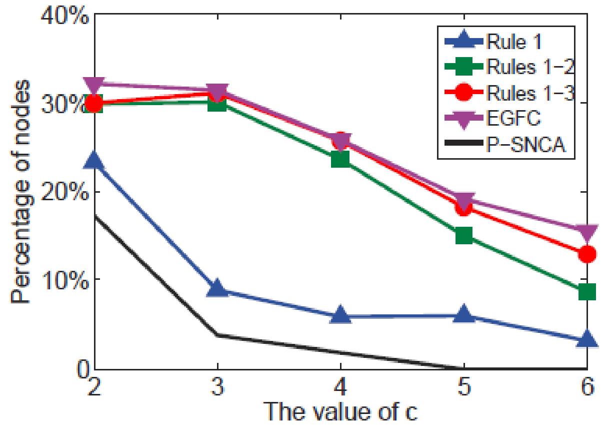


(b) Ave. length of awake time

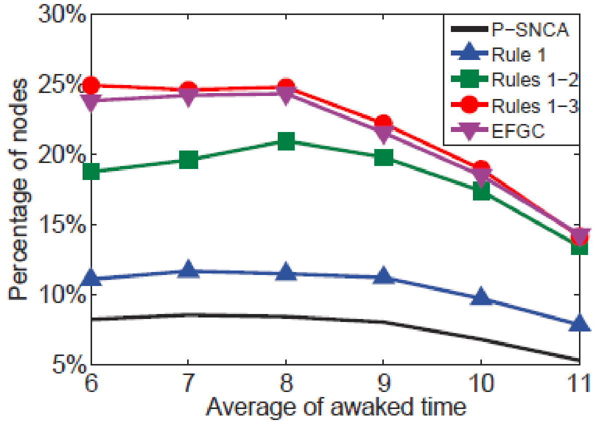


(c) Ave. length of sleep time

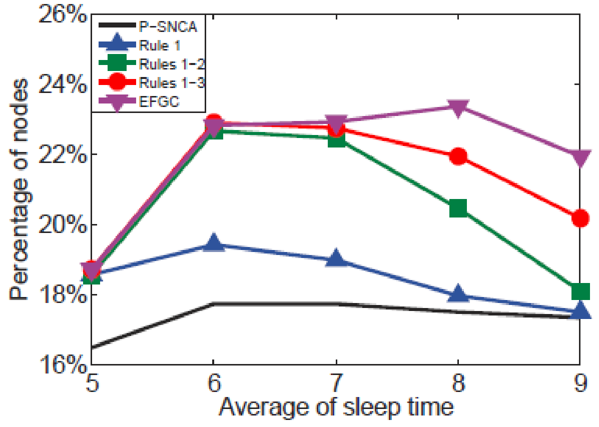
Evaluation



(a) Return requirement c



(b) Ave. length of awake time



(c) Ave. length of sleep time

Conclusion

- We first proposed a new crowdsourcing system, called social crowdsourcing, which explores the social relationships among workers.
- We considered the reliability issues on the social crowdsourcing system.
- We proposed GFC-structure and second-requester-based backup returning paths.
- We also provided 3 sets of rules for using these backup returning paths.

Thanks