institute IMdea networks

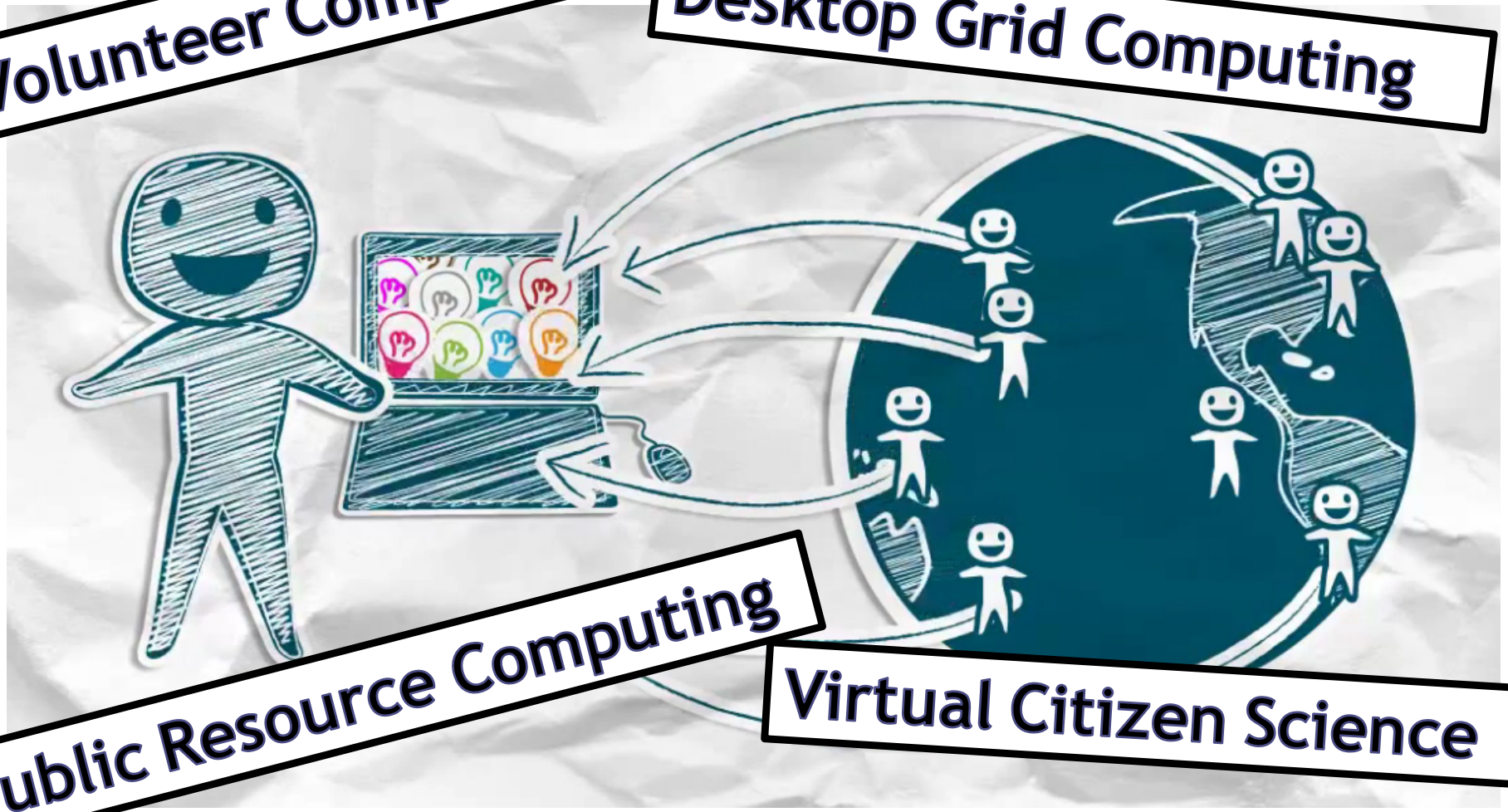
Universidad Carlos III de Madrid

Evaluating Reliability Techniques in the Master-Worker Paradigm

Evgenia Christoforou	IMDEA Networks Institute and Univ. Carlos III de Madrid
Antonio Fernández Anta	IMDEA Networks Institute
Kishori M. Konwar	Massachusetts Institute of Technology (MIT)
Nicolas Nicolaou	IMDEA Networks Institute

Developing the Science of Networks

Motivation

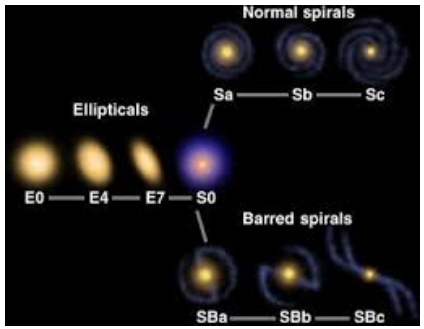

Volunteer Computing
Desktop Grid Computing
Public Resource Computing
Virtual Citizen Science

Examples


SETI@home
The Search for Extraterrestrial Intelligence
BOINC
Task
Result
Task
Result
Result
Task
Task
Result
Task
Result

Examples

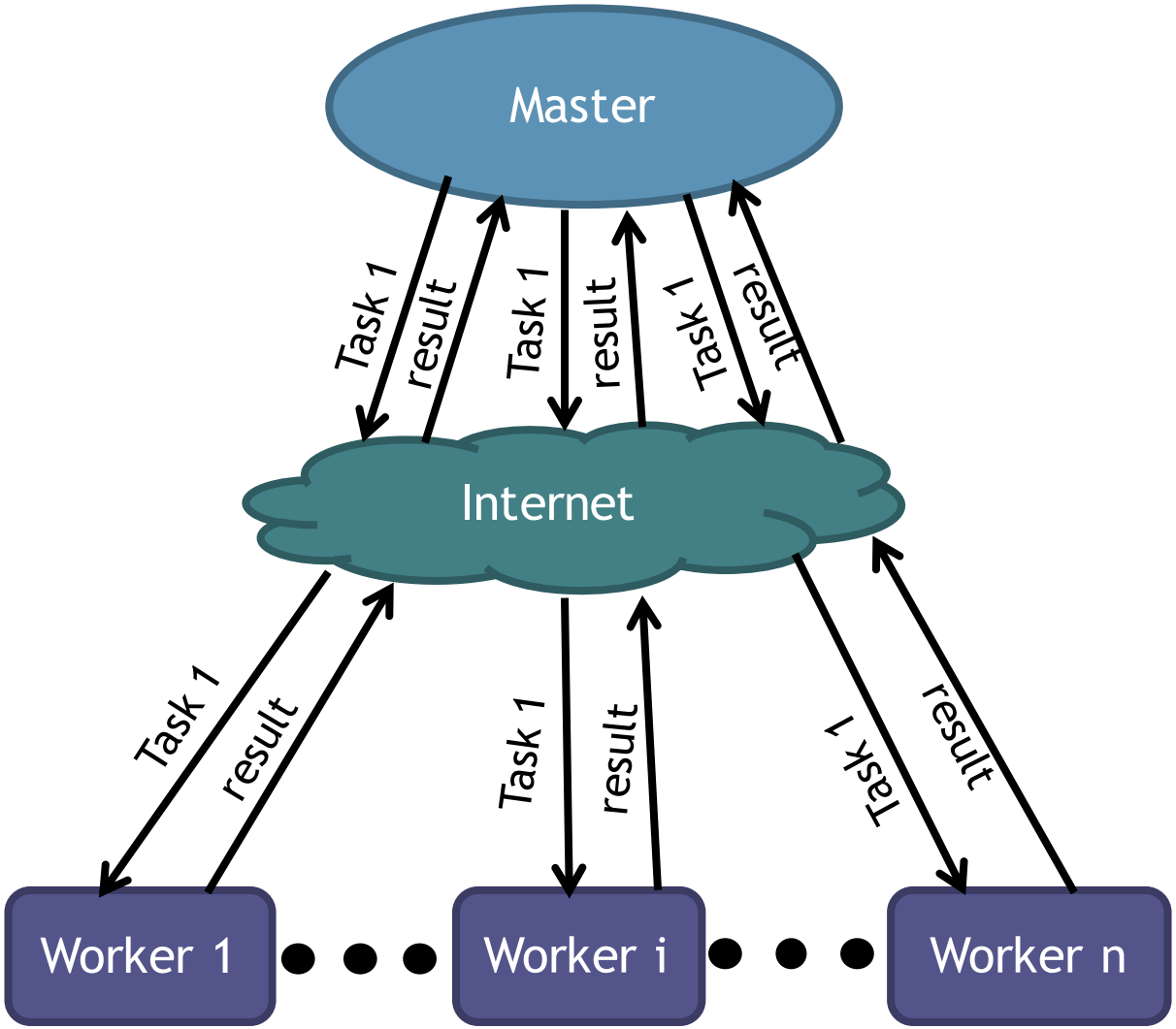
- Other projects running on the BOINC platform are related to issues regarding:
  - Health
  - Sustainability
  - Astronomy
- Other type of projects like Galaxy Zoo ask volunteers to:
  - Classify galaxies



  - And other complicated classification questions

by using their intelligence and report through the application their findings.

The master-worker paradigm



The Nature of the Workers

- Computation is carried out over the Internet, hence workers are untrustworthy, they can:
  - **Deliberately** provide incorrect results
  - Have hardware or software **failure** that corrupt the results
- We can classify the workers in two types:

Altruistic: Positive towards executing a task, willing to provide the correct result

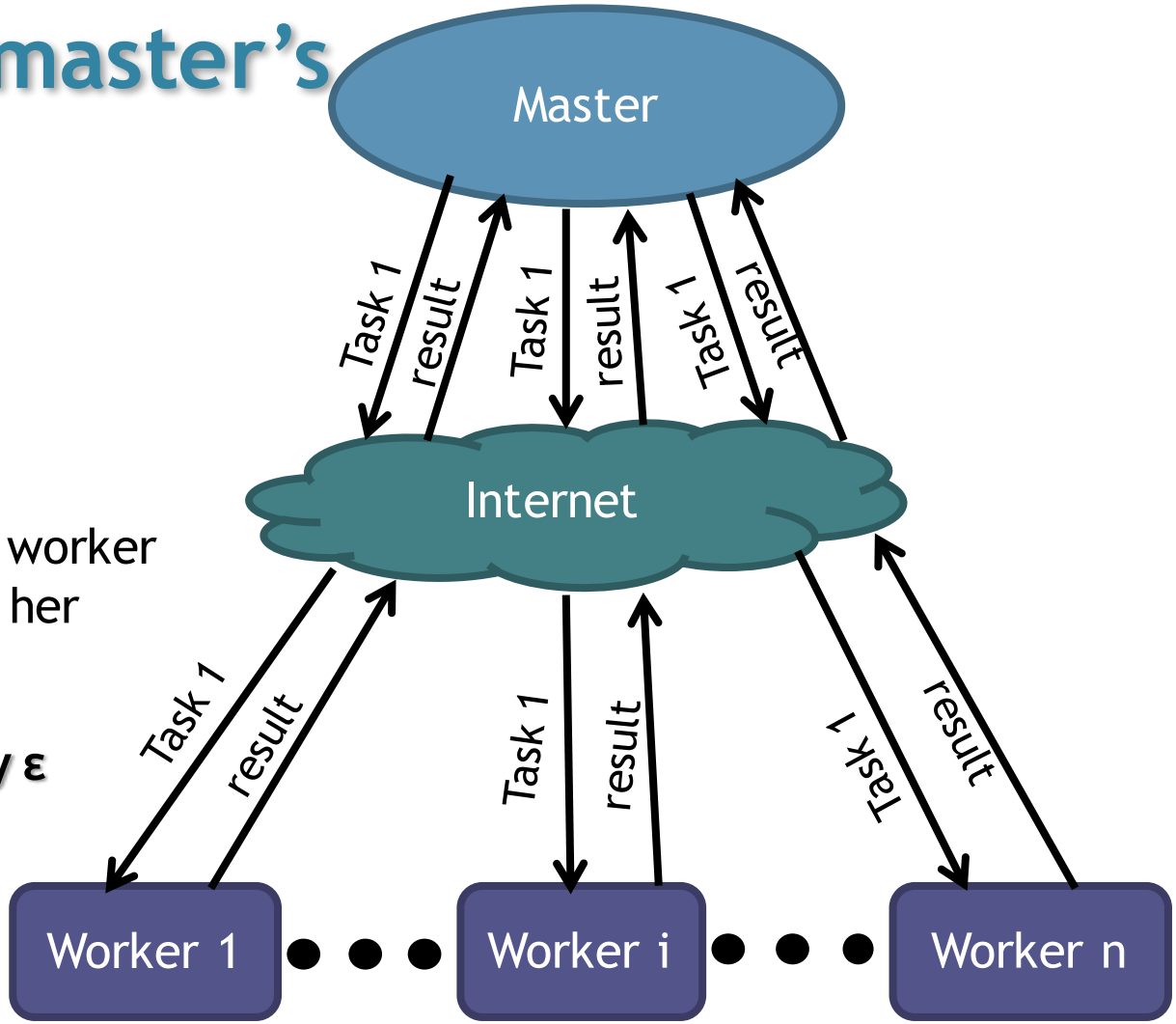
Troll: Negative towards executing a task, wants to convey an incorrect result

The Nature of the Workers

What are the master's "weapons"?

Either altruistic or troll a worker might fail to comply with her intended behavior

error probability ε for each worker



The master's techniques

- The most popular techniques used in the literature to increase the reliability of the results are:
  - **Voting:** collect multiple results on the same task from different workers and use a voting technique to decide on the correct result
    - Cons: 1) high concentration of incorrect results might lead to a wrong decision, 2) assigning the same task to multiple workers adds an extra load to the computation
  - **Challenges:** master uses task with a known solutions to detect the altruistic workers
    - Cons: 1) a worker might reply an incorrect result while replying correctly to the challenge, 2) adding extra load to the computation by using resources (workers) for known solutions, 3) the execution time increases

Related Work

Previous works assumed the presence of altruistic and malicious workers under different assumptions:

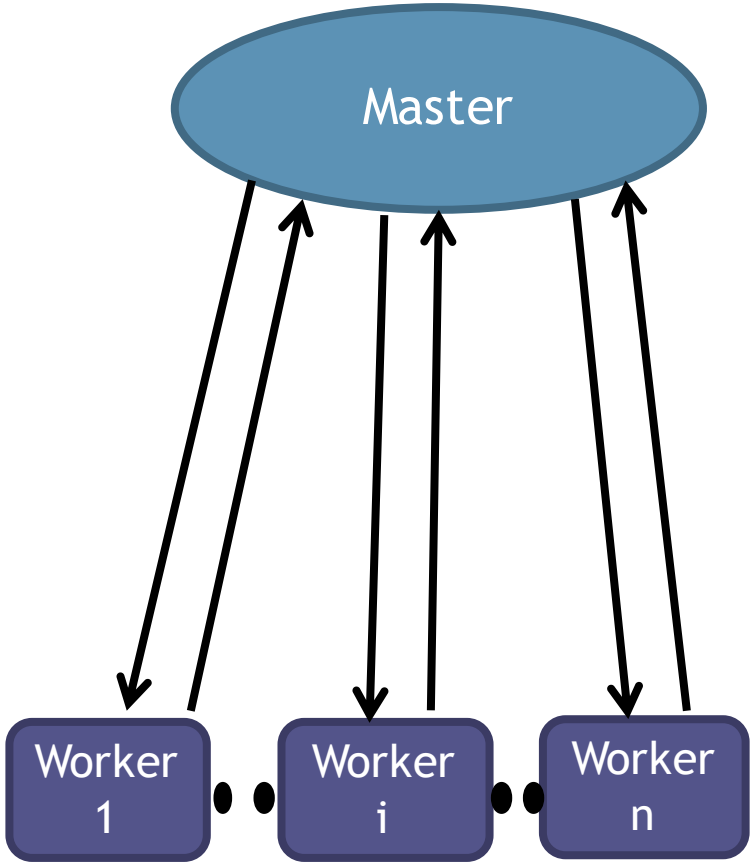
- Fernández et al., presented two voting mechanisms under the assumptions that the number of malicious workers or the workers probability of acting maliciously is known
- Konwar et al. do not make any assumptions on malice but rather try to approximate the probability of a worker being malicious
- Sarmenta assumed that only malicious workers have a constant probability of submitting an erroneous result
- Zhao and Lo compared voting and challenges under two assumptions, that malicious return the same incorrect result or that return different incorrect results. This work was mostly experimental
- Sonnek et al. designed algorithms for efficient task allocation based on the reputation of each worker. The algorithms proposed where evaluated through simulations

None of these works assumes a density of solutions

Contributions

- We model the master-worker paradigm in the presence of **altruistic** and **troll** workers using **5 system parameters**
- We define two measures for evaluating the algorithms complexity: **"time"** and **"work"**
- We assume **error probability ε=0** and we evaluate the two techniques. We show a negative result in the case of voting where the probability of receiving the correct answer is smaller than one.
- We assume **ε>0** and we present two algorithms, one that uses challenges and voting and another one that uses only voting. Both algorithms assume that certain system parameters are known
- Finally we present an algorithm to **estimate** some of the system parameters.

Model



Model

Problem Statement: The master must guarantee with high probability the correct result for each task $t_i \in T = \{t_i, \ldots, t_n\}$, without computing the task locally

Communication round for process *p*:

i. Receive message

ii. Perform computation &produce message

iii. Send result

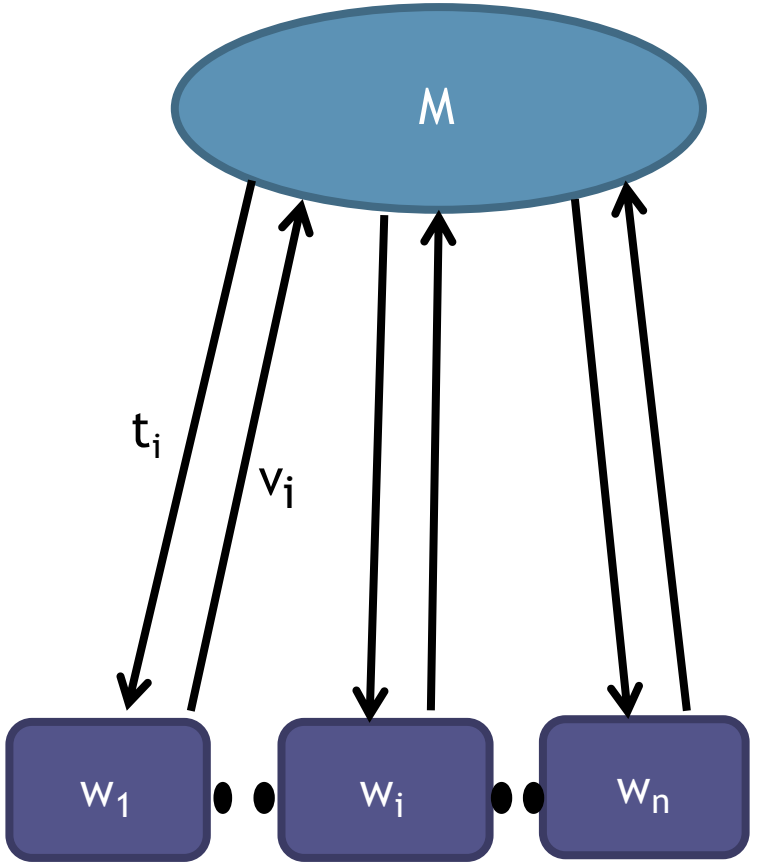
Performance Measures:

i. Time: the number of rounds needed by the algorithm to determine the result of *n* tasks

ii. Work: the number of aggregated results computed by each worker in the algorithm

Worker Type:

- $\mathbf{W_a}$ the set of altruistic workers, $n_a = |W_a|$, $n_a \geq 1$, $f_a = n_a / n$
- $\mathbf{W_t}$ the set of troll workers, $n_t = |W_t|$



$W = \{w_1, .. w_n\}$, $|W| = n$

ε , is the error probability

Model

Result Evaluation:
i. **Challenges (*C*)**
ii. **Voting (*V*)**

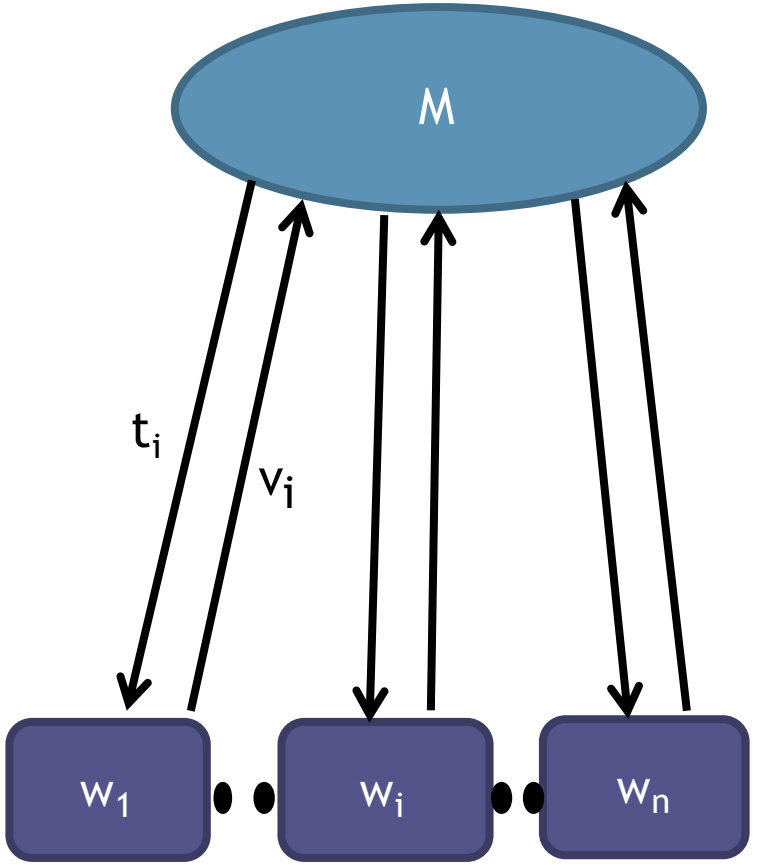
Density of Solutions:
• a reported result takes values from the D(t) domain
• correct solutions set $S(t) \subset D(t)$ for task t
• incorrect solutions set $R(t) = D(t) \setminus S(t)$ for task t
We assume that $d = |D(t)|, s = |S(t)|, r = |R(t)|$ are the same for every $t \in T$

$\frac{s}{d}$ **is the density of solutions for every task**

Environmental Parameters:

$(\varepsilon, s, r, f_a, T)$

worker error probability
number of incorrect replies
number of correct replies
results evaluation techniques
fraction of altruistic workers



Exact Worker Behavior (ϵ=0)

Fact: any algorithm needs at least $\frac{n}{n_a}$ time and needs n amount of work to compute correctly $|T| = n$ tasks with full reliability

Algorithm 1 Simple algorithm MWSIMPLE_0 where $\epsilon = 0$ and $T = \{C\}$.

1: Send challenge task t to all workers in W
2: $R[j] \leftarrow$ result received from $w_j \in W$, $j \in [1, |W|]$
3: $U_a \leftarrow \{w_i | R[i]$ is correct$\}$
4: **for** $i = 1 : |U_a| : n$ **do** ▷ for loop increments i by $|U_a|$
5: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
6: Add received result for t_{i+k-1} into $Results[i + k]$
7: **end for**
8: **return** $Results$

$\frac{n}{n_a}$ rounds

Theorem 3.1: Algorithm MWSIMPLE_0 has asymptotically optimal time $\Theta(\frac{n}{n_a})$ and optimal work $\Theta(n)$, and compute all the n tasks with probability 1, when $\epsilon = 0$.

Exact Worker Behavior (ϵ=0)

- If $n_a > s \cdot n_t$, then no incorrect value can appear more than n_t times.
- From the pigeonhole principle at least one correct value appears at least $n_a / s > n_t$ times.
- Thus every worker that returns values that appeared more than n_t times is altruistic.

Algorithm 2 Simple algorithm MWVOTE_0 where $\epsilon = 0$, $n_a > s \cdot n_t$, and $T = \{V\}$.

1: Send task t_1 to all workers in W
2: Add worker w_j to set $R[v]$ if it replied with value v
3: $U_a \leftarrow \bigcup_{v:|R[v]|>n_t} R[v]$
4: $Results[1] \leftarrow$ any value $v : |R[v]| > n_t$
5: **for** $i = 2 : |U_a| : n$ **do** ▷ loop increments i by $|U_a|$
6: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
7: Add received result for t_{i+k-1} into $Results[i + k - 1]$
8: **end for**
9: **return** $Results$

identifying at least $n_a - n_t(s-1) > n_t$ altruistic workers

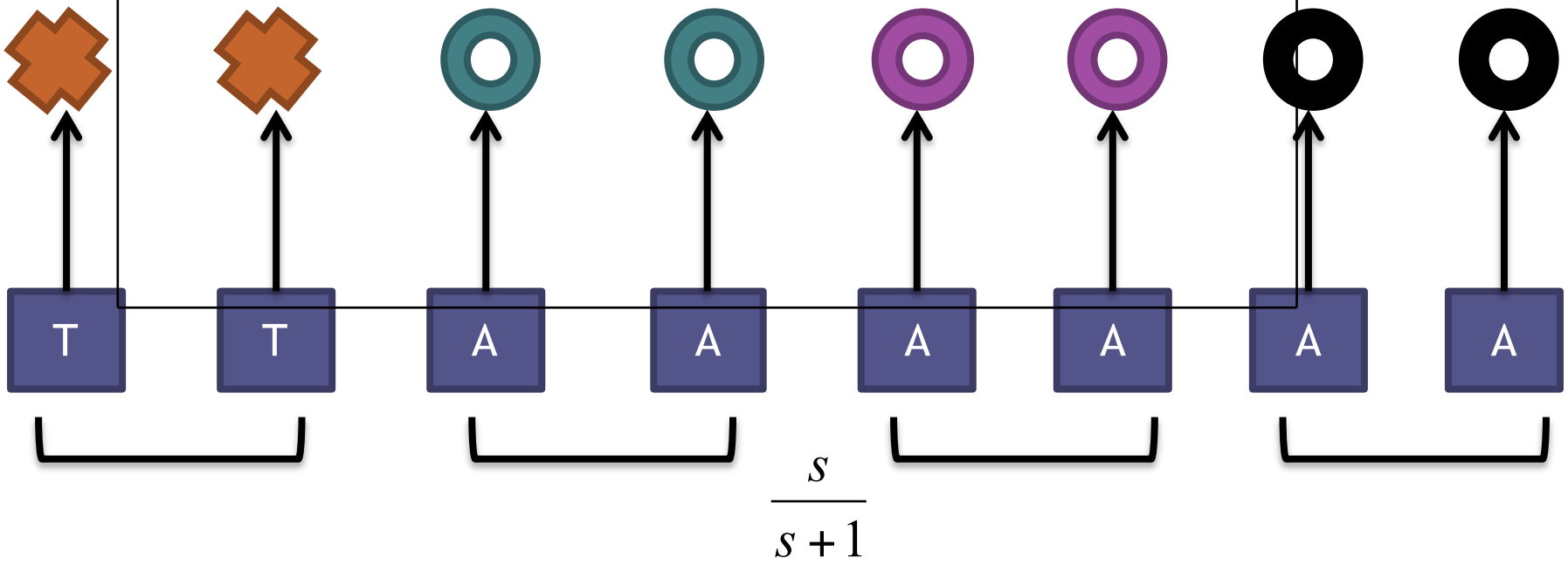
at most $\frac{n-1}{n_a - n_t(s-1)}$ rounds

Theorem 3.2: The algorithm MWVOTE_0 compute all the n tasks with probability 1 when $\epsilon = 0$ and $n_a > s \cdot n_t$. It has time $O(\frac{n}{n_t})$ and optimal work $\Theta(n)$.

Exact Worker Behavior (ϵ=0) (Negative Result)

Theorem 3.3: If $\epsilon = 0$ and $n_a = s \cdot n_t$, then for any $r > 0$ there exists no algorithm that allows the master node to returns the correct result of a task t with probability greater than $\frac{s}{s+1}$ in any execution.

Example: $n_a = 6, n_t = 2, s = 3, r = 1$



Probabilistic Worker Behavior (0<ϵ<1/2) Algorithm using challenges and voting

Algorithm 3 The pseudo-code for algorithm MWMIX, at the master, with n workers W computing the results of n tasks in $\mathcal{T}$, where $\frac{s}{s+1} < 1 - \epsilon$ and $T = \{C, V\}$.

Phase 1

1: $R[1..n] \leftarrow \emptyset^n$ ▷ $R[j]$ is the list of results from worker w_j
2: **for** $i = 1 : \lceil c \log n \rceil$ **do**
3: Send challenge task t to all workers in W
4: Add received result from worker w_j to $R[j]$
5: **end for**
6: **for** $i = 1 : n$ **do**
7: **if** # correct results in $R[i] \geq \lceil \frac{1}{2} c \log n \rceil$ **then**
8: $U_a \leftarrow U_a \cup \{w_i\}$
9: **end if**
10: **end for**

Phase 2

11: $F[i] \leftarrow \emptyset$ ▷ initially empty for all $1 \leq i \leq n$
12: **for** $j = 1 : \lceil k \log n \rceil$ **do**
13: **for** $i = 1 : |U_a| : n$ **do** ▷ loop increments i by $|U_a|$
14: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
15: Add received result for t_{i+k-1} to $F[i + k - 1]$
16: **end for**
17: **end for**
18: **for** $i = 1 : n$ **do**
19: $Results[i] \leftarrow plurality(F[i])$
20: **end for**
21: **return** $Results$

Finding the altruistic workers

Each task n is executed klogn times by the workers selected as altruistic

Probabilistic Worker Behavior (0<ϵ<1/2) Algorithm using challenges and voting

Algorithm 3 The pseudo-code for algorithm MWMIX, at the master, with n workers W computing the results of n tasks in $\mathcal{T}$, where $\frac{s}{s+1} < 1 - \epsilon$ and $T = \{C, V\}$.

Phase 1

1: $R[1..n] \leftarrow \emptyset^n$ ▷ $R[j]$ is the list of results from worker w_j
2: **for** $i = 1 : \lceil c \log n \rceil$ **do**
3: Send challenge task t to all workers in W
4: Add received result from worker w_j to $R[j]$
5: **end for**
6: **for** $i = 1 : n$ **do**
7: **if** # correct results in $R[i] \geq \lceil \frac{1}{2} c \log n \rceil$ **then**
8: $U_a \leftarrow U_a \cup \{w_i\}$
9: **end if**
10: **end for**

Phase 2

11: $F[i] \leftarrow \emptyset$ ▷ initially empty for all $1 \leq i \leq n$
12: **for** $j = 1 : \lceil k \log n \rceil$ **do**
13: **for** $i = 1 : |U_a| : n$ **do** ▷ loop increments i by $|U_a|$
14: Send task t_{i+k-1} to kth worker in U_a, $k \in [1, |U_a|]$
15: Add received result for t_{i+k-1} to $F[i + k - 1]$
16: **end for**
17: **end for**
18: **for** $i = 1 : n$ **do**
19: $Results[i] \leftarrow plurality(F[i])$
20: **end for**
21: **return** $Results$

Finding the altruistic workers

Each task n is executed klogn times by the workers selected as altruistic

Lemma 4.1: In any execution of MWMIX, at the end of Phase 1 we have $U_a = W_a$, whp.

Theorem 4.1: If $\frac{s}{s+1} < 1 - \epsilon$, then Algorithm MWMIX computes all n tasks correctly, whp.

Theorem 4.2: Algorithm MWMIX runs in $\Theta(\frac{n}{n_a} \log n)$ synchronous rounds and performs $\Theta(n \log n)$ work.

Probabilistic Worker Behavior (0<ϵ<1/2) Algorithm using only voting

Algorithm 4 Algorithm MWVOTE, at the master process, performs n tasks using n workers for the case $\frac{s}{s+1} < f_a(1-\epsilon) + (1-f_a)\epsilon$ and $T = \{V\}$.

```
1: F[i] ← ∅                                   ▷ initially empty for all 1 ≤ i ≤ n
2: for i = 1 to ⌈k log n⌉ do                  ▷ for some constant k > 0
3:       Choose a random permutation π ∈ Π_n
4:       Send each task t_j ∈ T to worker w_π(j)
5:       Add received result from worker w_π(j) to F[j]
6: end for
7: for i = 1 : n do
8:     Results[i] ← plurality(F[i])
9: end for
10: return Results
```

Theorem 4.3: If $\frac{s}{s+1} < f_a(1-\epsilon) + (1-f_a)\epsilon$, Algorithm MWVOTE computes all n tasks correctly *whp*.

Theorem 4.4: Algorithm MWVOTE runs in $\Theta(\log n)$ synchronous rounds and performs $\Theta(n \log n)$ work.

Estimating f_a and ε

- We need to know f_a , ε and s to be able to choose and apply the latest two algorithms
- We can assume that s can be known, given that the master gives the task (e.g. Galaxy Zoo)
- We can estimate f_a and ε
  - we use user defined bounds in a manner called (ε, δ)-approximation
  - Choose $\varepsilon, \delta \in O(\frac{1}{n^c})$ for some c>0, in such a way that the estimate value is within a $\pm\varepsilon$ factor and with a probability $1-\delta$
  - Base on the stopping rule algorithm of Dagum et al.

Estimating f_a and ε

Algorithm 6 Algorithm E_1 to estimate f_a, ϵ, and $f_a(1-\epsilon)+(1-f_a)\epsilon$.

1: Let $\delta = \frac{1}{n^c}$ and $\varepsilon = \frac{1}{n^c}$ for $c > 0$
2: Let $\Gamma = (4\lambda \log(\frac{2}{\delta}))/\varepsilon^2$ and $\Gamma_1 = 1 + (1+\varepsilon)\Gamma$
3: Let $\ell = \lceil k \log n \rceil$, for some $k > 0$
4: $N \leftarrow 0, S \leftarrow 0$
5: **while** $S < \Gamma_1$ **do**
6: $N \leftarrow N + 1$
7: pick a worker w randomly uniformly from W
8: **for** $i = 1$ to ℓ **do**
9: send challenge task t_i to w
10: $R[i] \leftarrow$ result received from w
11: **end for**
12: **if** $CorrMaj(R)$ **then** $Z_N^1 \leftarrow 1$ **else** $Z_N^1 \leftarrow 0$ **end if**
13: $S \leftarrow S + Z_N^1$
14: **end while**
15: $\tilde{p} \leftarrow \frac{\Gamma_1}{N}$
16: $N \leftarrow 0, S \leftarrow 0$
17: **while** $S < \Gamma_1$ **do**
18: $N \leftarrow N + 1$
19: pick a worker w randomly uniformly from W
20: send challenge task to w
21: **if** result received from w is correct **then** $Z_N^2 \leftarrow 1$ **else** $Z_N^2 \leftarrow 0$ **end if**
22: $S \leftarrow S + Z_N^2$
23: **end while**
24: $\tilde{q} \leftarrow \frac{\Gamma_1}{N}$
25: **return** $\left(\tilde{p}, \frac{\tilde{q}-\tilde{p}}{1-2\tilde{p}}, \tilde{q}\right)$

f_a estimate

$f_a(1-\varepsilon)+(1-f_a)\varepsilon$ estimate

ε estimate

Theorem 5.3: The number of rounds or the work for algorithm E_1 is $n^c \log n$ for $c > 0$ *whp*.

Conclusions and Future Work

- We considered the master-worker paradigm to model Internet-based task computations in the presence of altruistic and troll workers
  - We assumed that workers could deviate from their true behavior based on an error probability
  - We considered tasks with that can have multiple correct and multiple incorrect solutions

In the **future** we plan to explore the following aspects:

- possible unavailability of the workers
- each worker might have a different error probability
- workers might have different error probabilities over time
- tasks with different number of correct and incorrect results

We believe that the above improvements to the considered model will allow us to capture the **crowdsourcing** paradigm

evgenia.christoforou@imdea.org
Evgenia Christoforou
PhD Student @ IMDEA Networks Institute, Madrid, Spain