UNIVERSITY OF LIVERPOOL

LEVERHULME TRUST

Simple and Fast Approximate Counting and Leader Election in Populations

Michail Theofilatos

Joint work with
Othon Michail and Paul G. Spirakis

Department of Computer Science, University of Liverpool, UK
Computer Engineering and Informatics Department, University of Patras, Greece

20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2018)
November 4-7, 2018
Tokyo, Japan

The Leverhulme Research Centre for Functional Materials Design

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

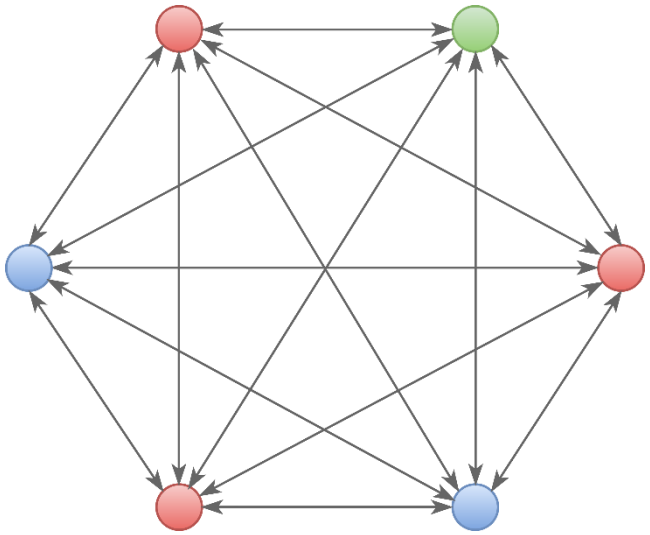


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler.*
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

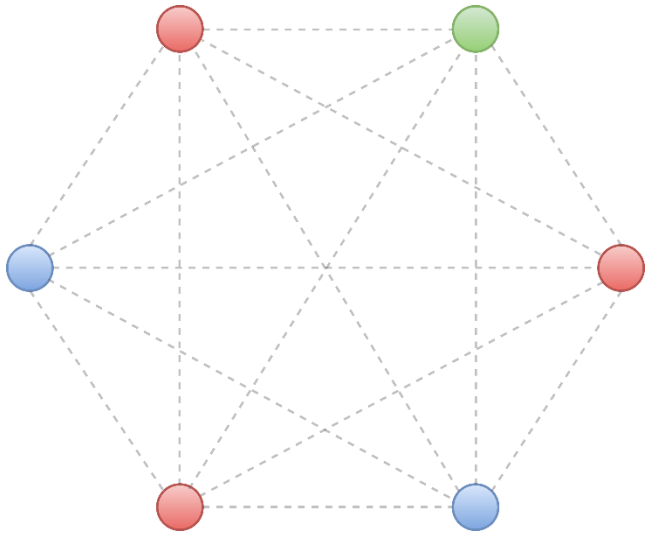


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

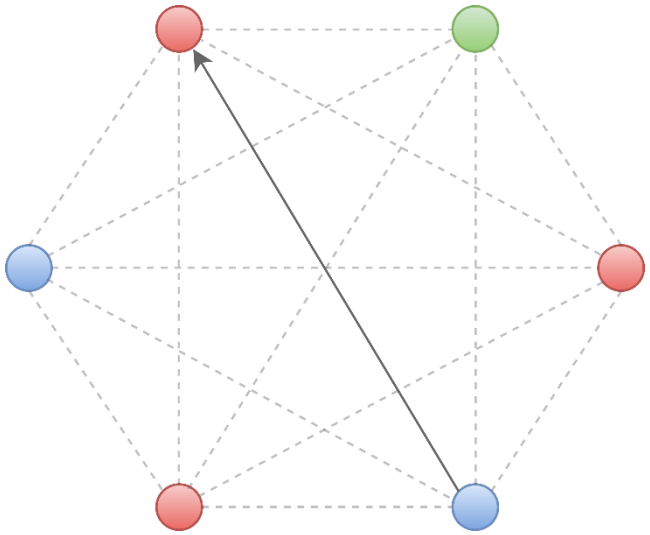


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

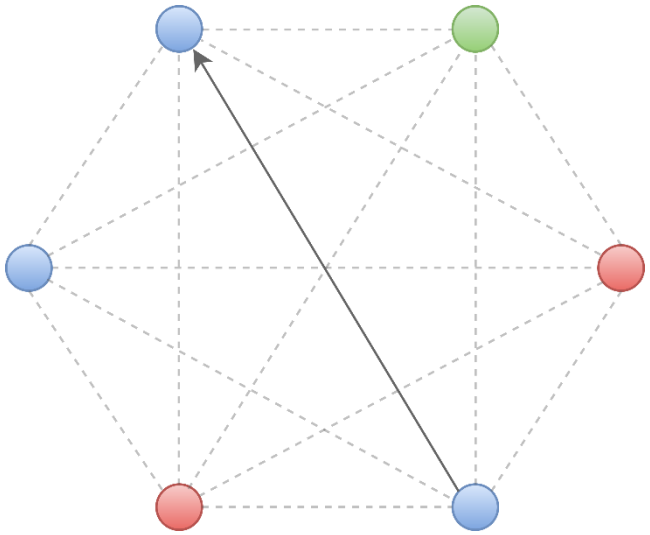


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

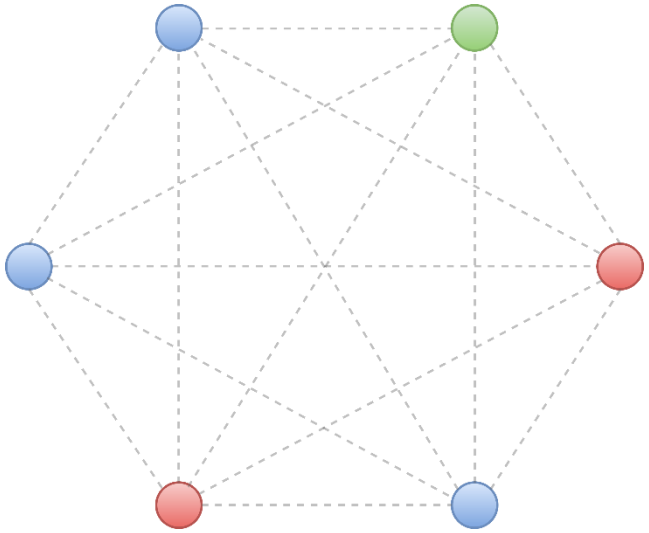


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

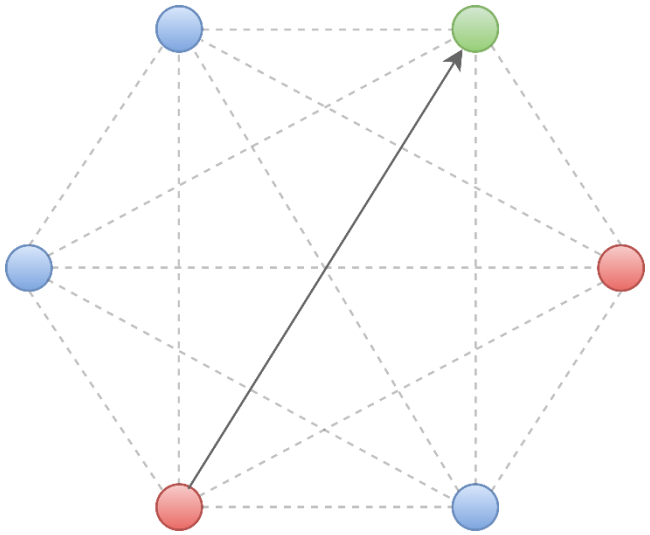


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

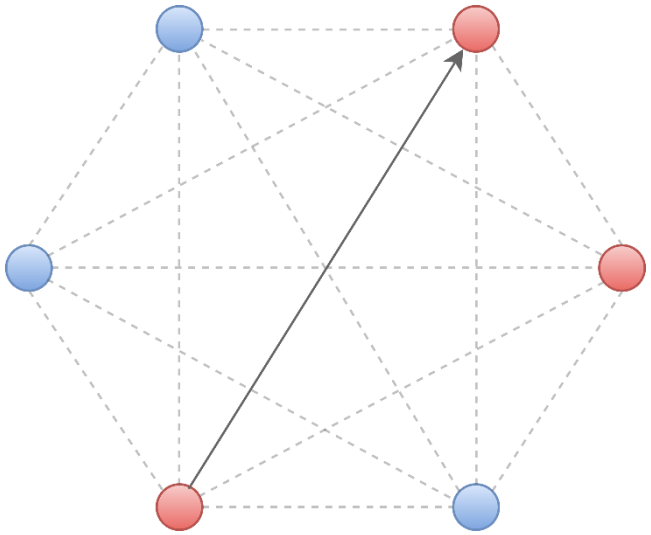


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Population Protocols

- Distributed computing model formed by n resource-limited mobile agents
  - Interact in pairs
  - Cannot control their interactions
    - Passive mobility, like particles in a well-mixed solution.
    - A pair of agents interacts in every discrete step under a *uniform random scheduler*.
    - Complete communication graph $G = (V, E)$
- Anonymous (i.e., do not have unique IDs) agents.
- Finite set of states Q.
- Transition function $\delta: Q \times Q \rightarrow Q \times Q$.

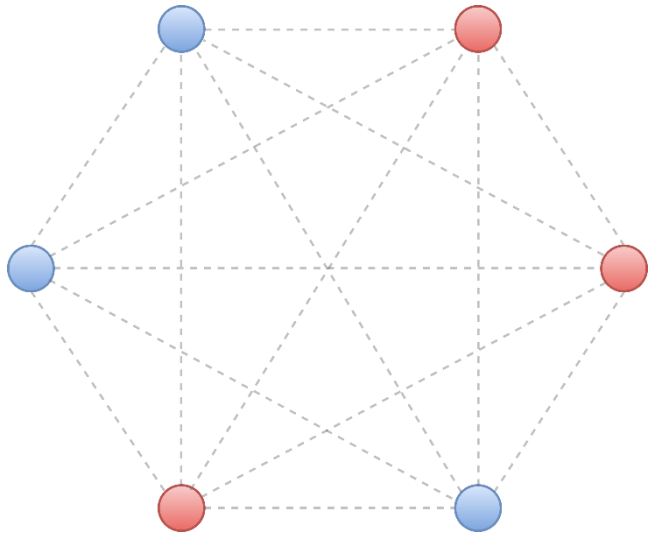


Figure 1: Complete Interaction graph. Each colour represents a different state.

[Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235–253, March 2006]

Related Work

- Approximate Counting: *not much is known*
  - [Michail, PODC, '15] computes an approximate count between *n/2* and *n* in $O(\text{nlogn})$ parallel time w.h.p.
- Leader Election:
  - Simple pairwise-elimination protocol requires linear parallel time [AADFP, Distr. Comp., '06].
  - Any *standard* population protocol requires linear parallel time to solve leader election [DS, DISC, '15].
  - If we strengthen the PP model, we can develop sub-linear time protocols.
    - [AG, ICALP, '15] stabilises in $O(\log^3 n)$ parallel time, assuming $O(\log^3 n)$ states at each agent.
    - [GSU, arXiv, '18] $O(\text{lognloglogn})$ parallel time, assuming $O(\text{loglogn})$ states.
    - [GS, SODA '18] O(log^2n) parallel time, assuming $O(\text{loglogn})$ states.
    - Mediated PP model: [MOKY, Distr. Comp. '12], [DLFSV, TAMC, '17]

Problems and Definitions

Approximate Counting Problem.

We define as *Approximate Counting* the problem in which a leader must determine an estimation $\hat{n}$ of the population size, where $\frac{\hat{n}}{a} < n < \hat{n}$.

Leader Election Problem.

Each node eventually decides whether it is a leader or not, subject to only one node decides that it is the leader.

One-way epidemic.

There is a single node in state *a*, *n-1* nodes in state q and the only effective transition is (a,q)→(a,a). The expected number of steps until all nodes become *a* is Θ(logn) parallel time.

Approximate Counting Protocol

- Requires a unique leader.
- Runtime: $\Theta(\log n)$ parallel time.
- $O(1)$ number of states, except for the unique leader ($O(\log^2 n)$ states).

Protocol 1 Population Size Estimation (PSE)

$Q = \{q, a, l_{c_0,c_1}\}$
δ :

$(l_{0,0}, q) \to (l_{1,0}, a)$
$(a, q) \to (a, a)$
$(l_{c_0,c_1}, q) \to (l_{c_0+1,c_1}, q),\ if\ c_0 > c_1$
$(l_{c_0,c_1}, a) \to (l_{c_0,c_1+1}, a),\ if\ c_0 > c_1$
$(l_{c_0,c_1}, \cdot) \to (halt, \cdot),\ if\ c_0 = c_1$

- The leader is in state *l*, and the rest are in state *q*.
- The leader initiates the epidemic (state *a*).
- The leader stores two counters c_q and c_a.
- Counts the number of interactions with *q* and *a* nodes.
- Termination condition: $c_q = c_a$

Approximate Counting Protocol

THEOREM 1.

Our Approximate Counting protocol gives a constant-factor approximation of $logn$ in $O(logn)$ parallel time w.h.p..

- **First Phase** – Number of infected nodes $[1, \frac{n}{2}]$:
  - c_q reaches $O(logn)$ w. h. p.
  - c_a is increased by a small constant number $(\approx log2)$ *(w.h.p. less that* $(O(\sqrt{logn}))$
  - The protocol does not terminate w.h.p. until more than half of the population has been infected.
- **Second Phase** – Number of infected nodes $(\frac{n}{2}, n]$:
  - c_q is increased by a small constant number $(\approx log2)$ *(w.h.p. less that* $(O(\sqrt{logn}))$
  - c_a eventually catches up c_q.
- 2^{c_q} is an upped bound on the population size.
- First phase: $\Theta(\text{logn})$ parallel time. Second phase: $\Theta(\text{logn})$ parallel time.

Approximate Counting Protocol

THEOREM 1.

Our Approximate Counting protocol gives a constant-factor approximation of $logn$ in $O(logn)$ parallel time w.h.p..

- **First Phase** – Number of infected nodes $[1, \frac{n}{2}]$:
  - c_q reaches $O(logn)$ w. h. p.
  - c_a is increased by a small constant number $(\approx log2)$ *(w.h.p. less that* $(O(\sqrt{logn}))$
  - The protocol does not terminate w.h.p. until more than half of the population has been infected.
- **Second Phase** – Number of infected nodes $(\frac{n}{2}, n]$:
  - c_q is increased by a small constant number $(\approx log2)$ *(w.h.p. less that* $(O(\sqrt{logn}))$
  - c_a eventually catches up c_q.
- 2^{c_q} is an upped bound on the population size.
- First phase: $\Theta(\text{logn})$ parallel time. Second phase: $\Theta(\text{logn})$ parallel time.

Leader Election Protocol

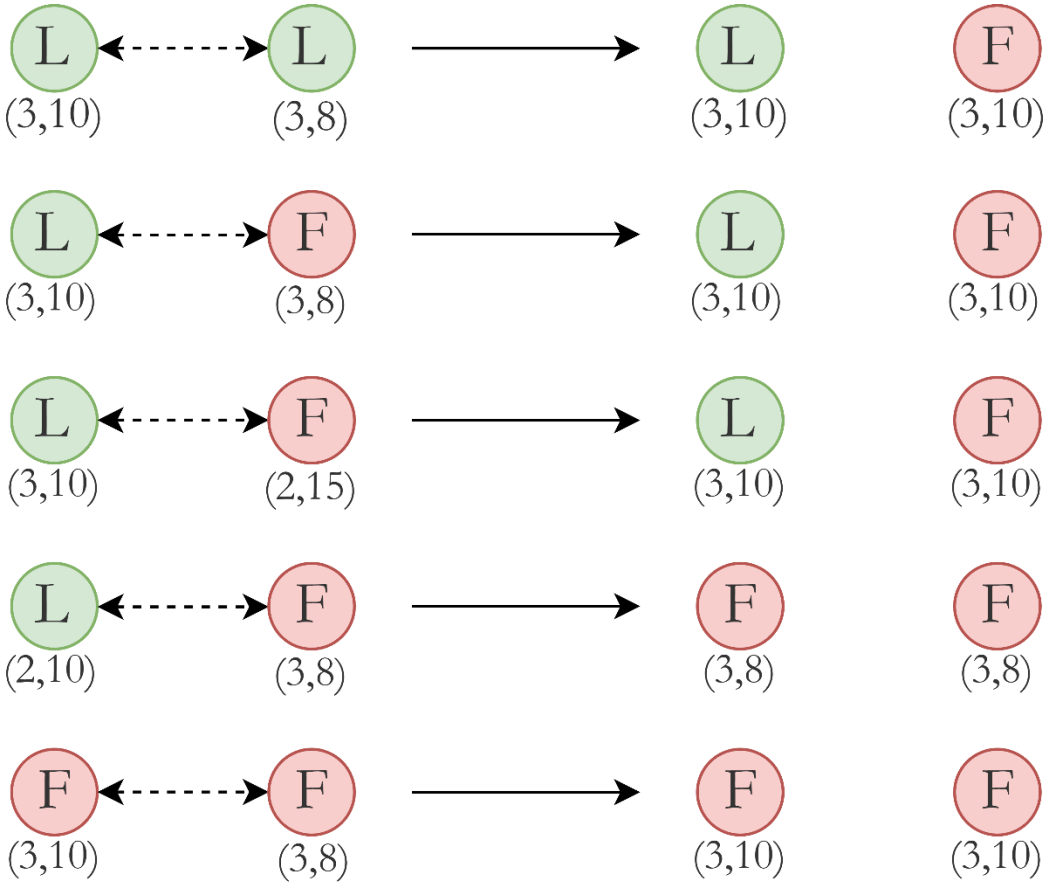
- Assumes that the nodes know an upper bound of $\log(n)$.
- They produce random numbers in $[1, m]$.
- Elects a unique leader in $O(\frac{\log^2 n}{logm})$ parallel time w.h.p..
- Initially all nodes are potential leaders (i.e., all nodes start from the same state).
- The protocol proceeds in rounds by monotonously reducing the set of possible leaders, until only one survives.

- All nodes store three variables: $Round, Random\ Number\ and\ Counter$.
- The *Counter* is increased by one in every interaction, as long as the node remains leader.
- When *Counter* reaches $blogn$, it resets it to zero and increases the $Round$ by one.
- They try to spread their tuple $(Round, Random\ Number)$ throughout the population.

Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

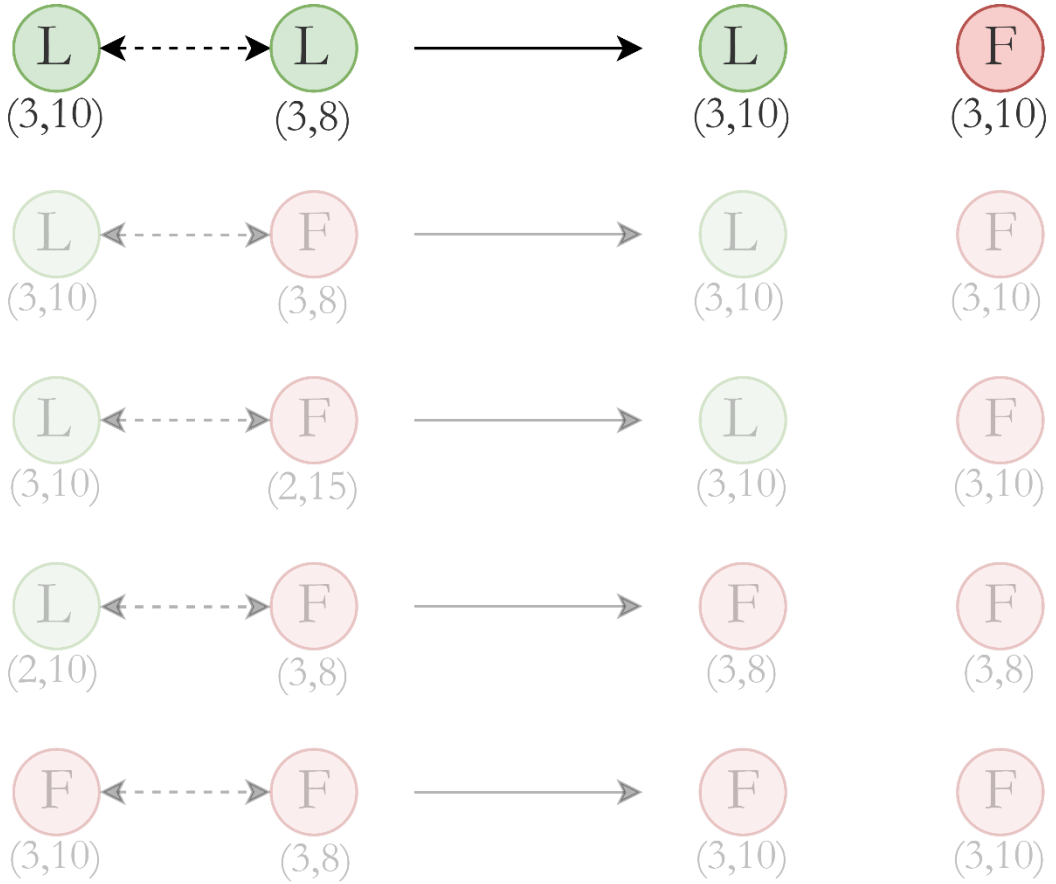
- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

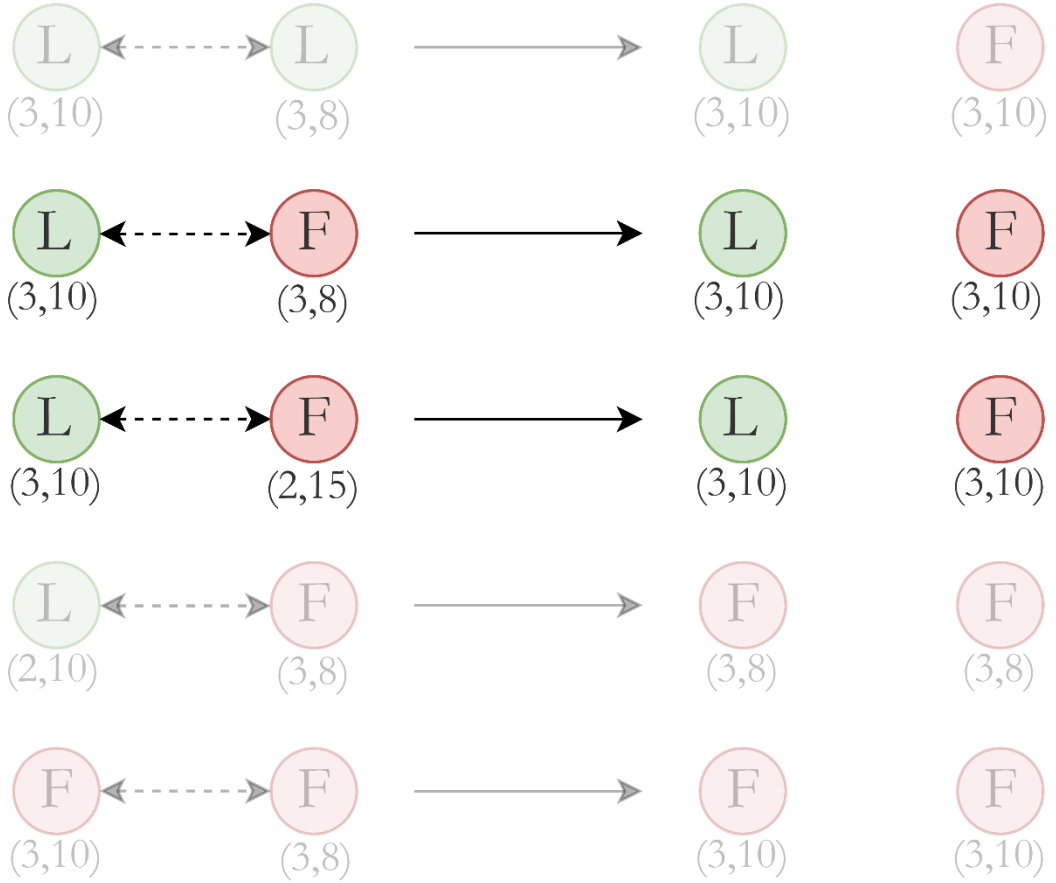
- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

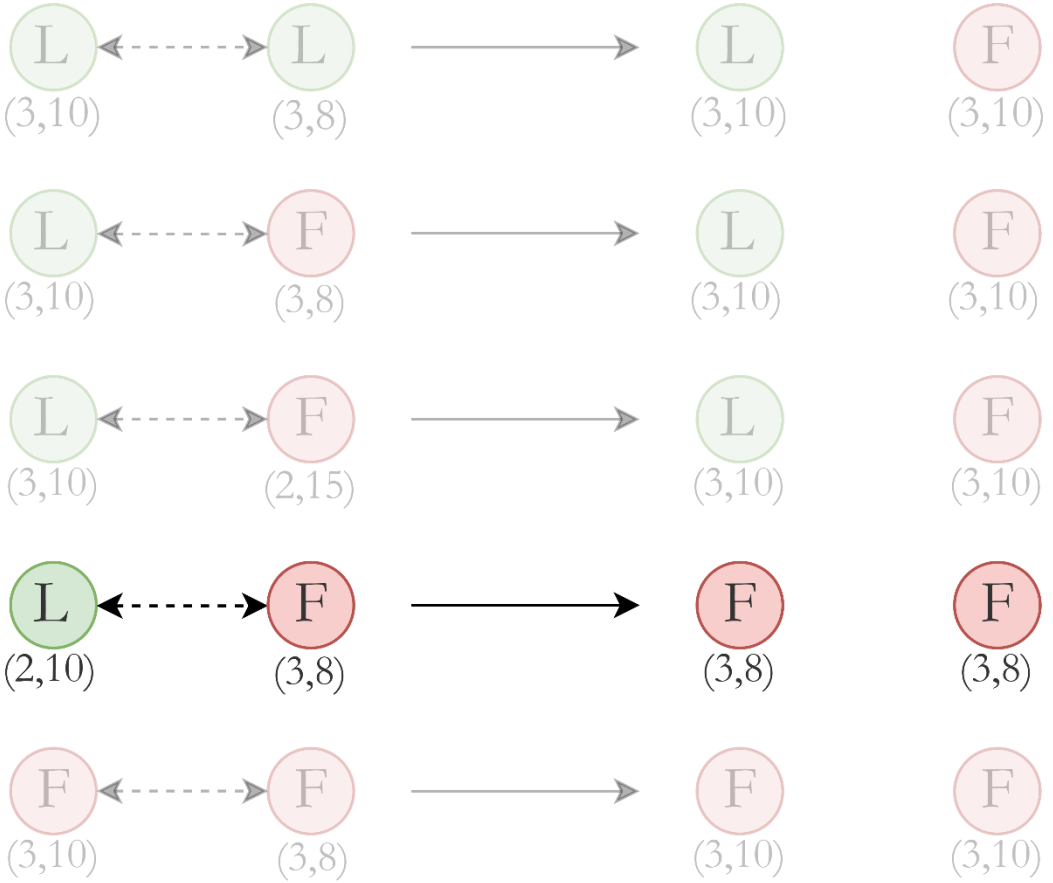
- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

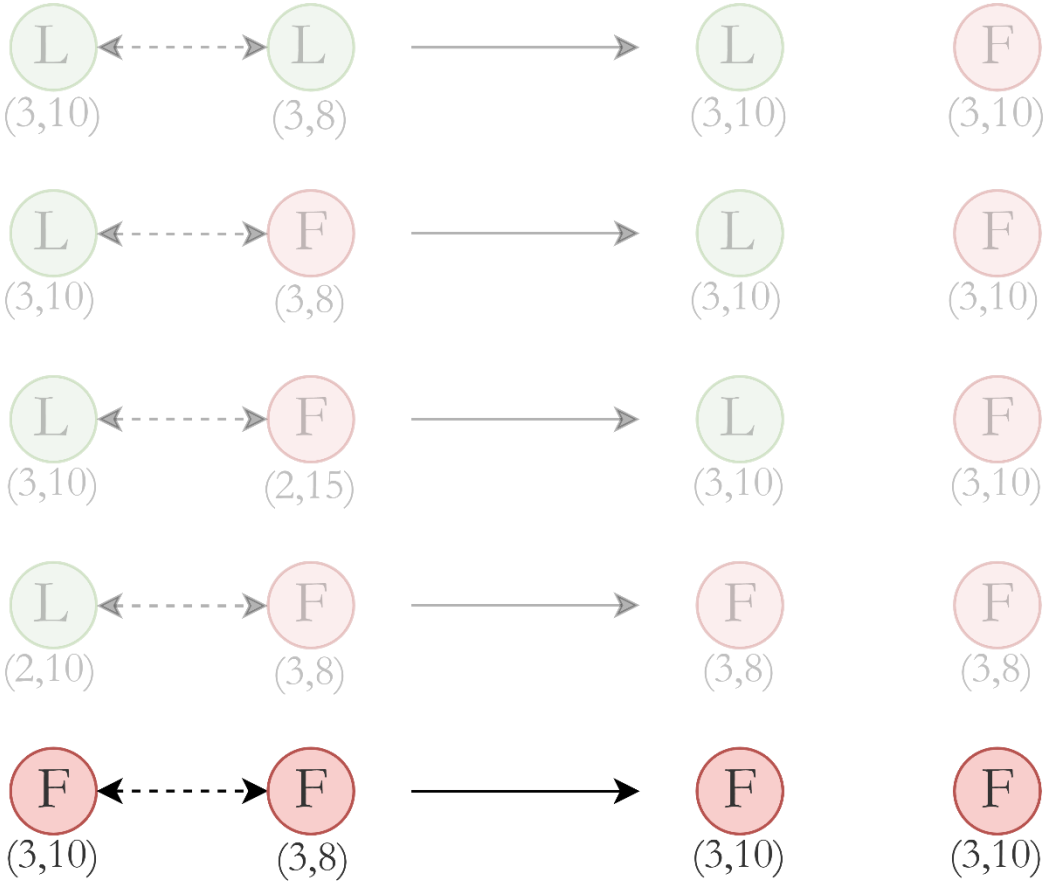
- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

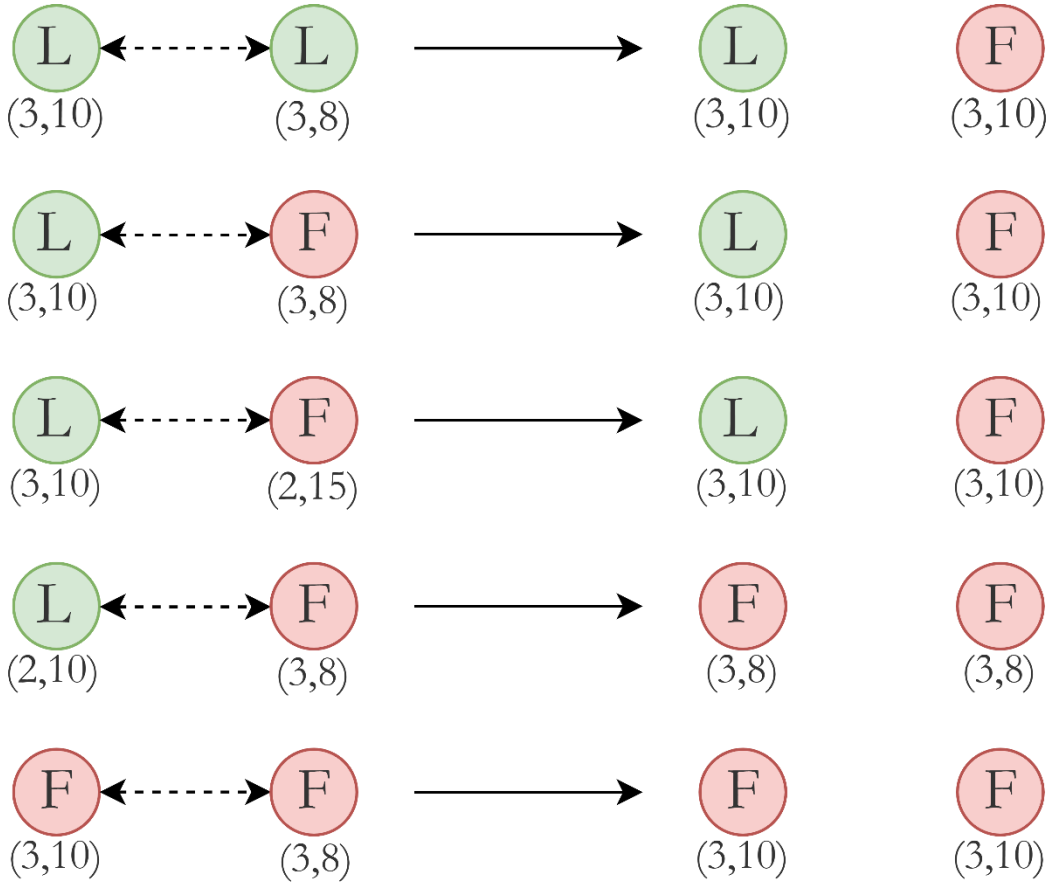
- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

- We say that a node with the tuple (e_1, r_1) wins during an interaction with (e_2, r_2) if:
  1. $e_1 > e_2$, *or*
  2. $e_1 = e_2$ *and* $r_1 > r_2$

- We call (e_1, r_1) the *dominant tuple* during the round e_1, if it wins every other tuple.



Leader Election Protocol

THEOREM 2.

Our Leader Election protocol elects a unique leader in $O(\frac{\log^2 n}{logm})$ parallel time w.h.p..

- During a round e, the dominant tuple is spread as an epidemic *(in $\Theta(logn)$ parallel time)*.
- No leader can enter the next round if its tuple has not been spread throughout the whole population before.
- At least one leader will always exist in the population.
- After $O(\frac{logn}{logm})$ rounds, a unique leader exists in the population w.h.p..

Leader Election Protocol

THEOREM 2.

Our Leader Election protocol elects a unique leader in $O(\frac{\log^2 n}{logm})$ parallel time w.h.p..

- During a round e, the dominant tuple is spread as an epidemic *(in $\Theta(logn)$ parallel time)*.
- No leader can enter the next round if its tuple has not been spread throughout the whole population before.
- At least one leader will always exist in the population.
- After $O(\frac{logn}{logm})$ rounds, a unique leader exists in the population w.h.p..

Composition of our protocols

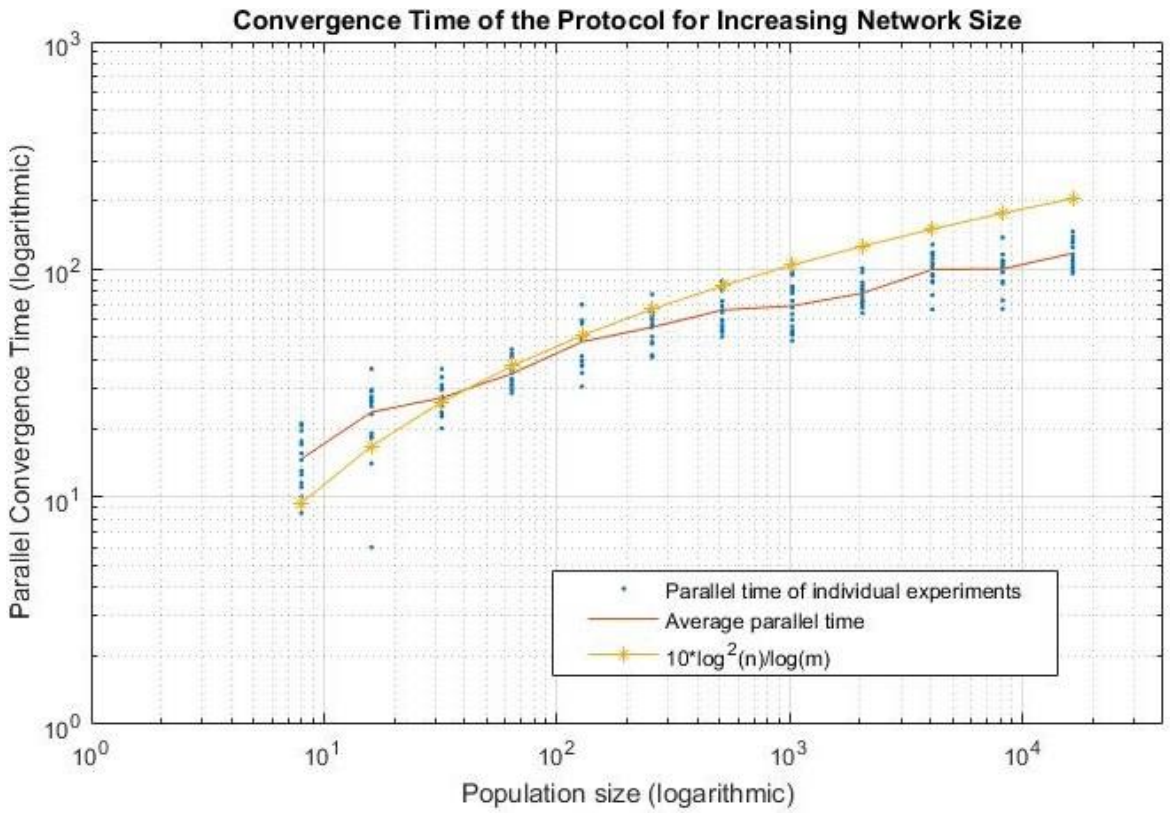
- Our leader election protocol requires a rough estimate on the size of the population
- Our approximate counting protocol requires a unique leader

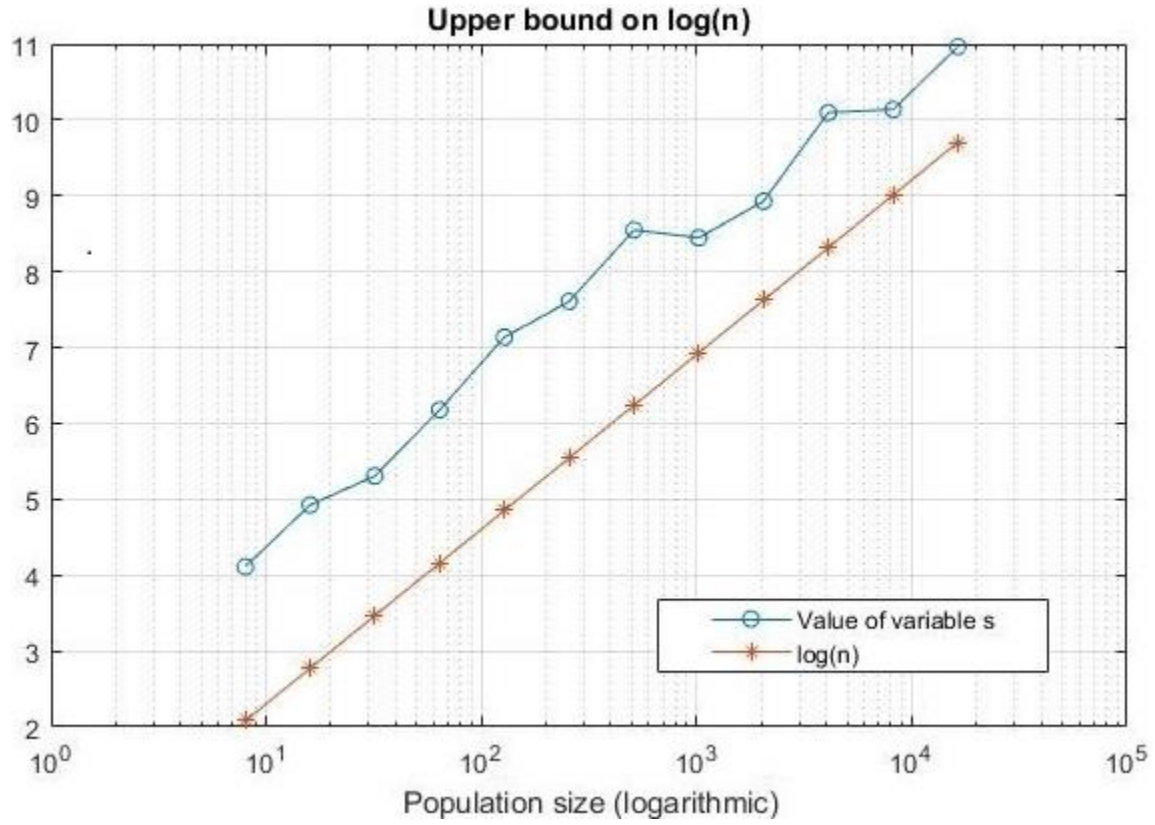
✓ We now combine our protocols in order to construct a leaderless and size-oblivious protocol which elects a unique leader and gives a constant factor approximation of $\log(n)$.

Composition of our protocols

- Leader Election and Approximate Counting protocol:
  - A. In the Leader Election protocol, instead of using one counter, use two counters c_q and c_a
    1. c_q: Counts the non-followers
    2. c_a: Counts the followers
  - B. When $c_q = c_a$
    1. Move to next round
    2. Reset counters ($c_a = 0$ and $c_q = 1$)
    3. Update the value of s as follows: $s' = \frac{s(e_1 - 1) + c_q}{e_1}$
    4. When $e_1 = \left\lceil \frac{as}{logm} \right\rceil$ stop increasing the round.

Composition of our protocols





Conclusions

- Approximate Counting protocol
- Leader Election protocol
- Composition of these protocols

- Can we design a *polylogarithmic* time population protocol that solves the leader election problem, if we allow the agents to communicate only constant amount of bits during an interaction?

UNIVERSITY OF LIVERPOOL

Thank You