Running-time Analysis of Broadcast Consensus Protocols

Philipp Czerner, Stefan Jaax
Fakultät für Informatik, TU München

March 23, 2021

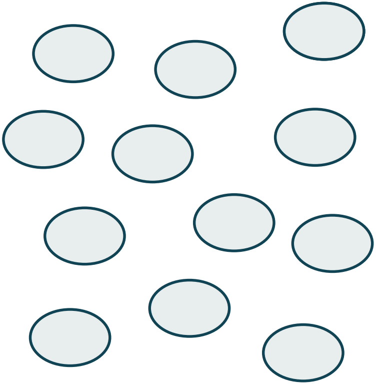
Introduction

The Setting

- distributed computation

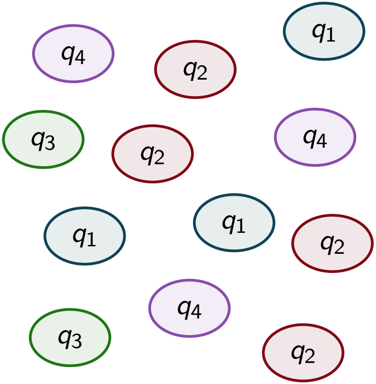
The Setting

- distributed computation
- population of *agents*



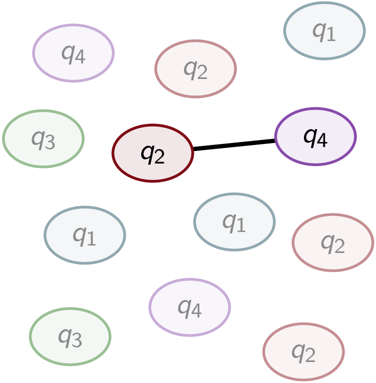
The Setting

- distributed computation
- population of *agents*
- agents are finite-state machines



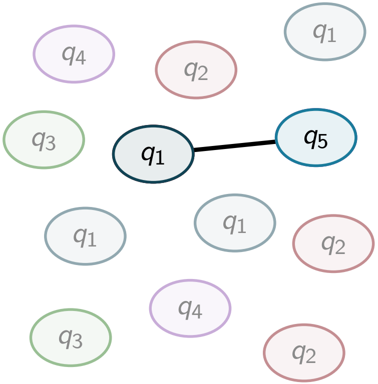
The Setting

- distributed computation
- population of *agents*
- agents are finite-state machines
- random interactions



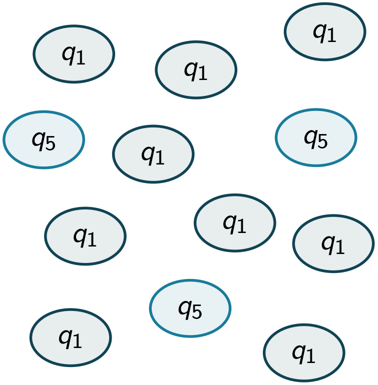
The Setting

- distributed computation
- population of *agents*
- agents are finite-state machines
- random interactions



The Setting

- distributed computation
- population of *agents*
- agents are finite-state machines
- random interactions
- want to reach consensus on whether the initial configuration satisfies a property



Population Protocols

- well-studied

Population Protocols

- well-studied
- finite set of states Q

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- compute exactly semi-linear (or Presburger) predicates

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- compute exactly semi-linear (or Presburger) predicates
  - i.e. predicates expressible in first-order theory of integers with addition and the usual order

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- compute exactly semi-linear (or Presburger) predicates
  - i.e. predicates expressible in first-order theory of integers with addition and the usual order
- can compute majority: $x \geq y$

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- compute exactly semi-linear (or Presburger) predicates
  - i.e. predicates expressible in first-order theory of integers with addition and the usual order
- can compute majority: $x \geq y$
  - $\Omega(n^2/\operatorname{polylog}(n))$ interactions to stabilise [Alistarh et al. 2017]

Population Protocols

- well-studied
- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- compute exactly semi-linear (or Presburger) predicates
  - i.e. predicates expressible in first-order theory of integers with addition and the usual order
- can compute majority: $x \geq y$
  - $\Omega(n^2/\operatorname{polylog}(n))$ interactions to stabilise [Alistarh et al. 2017]
  - $\mathcal{O}(n^{1+\varepsilon})$ interactions to converge [Kosowski, Uznański 2018]

Broadcasts Consensus Protocols

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
   - NL refers predicates decidable by log-space Turing machines, with input encoded as unary

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
    - NL refers predicates decidable by log-space Turing machines, with input encoded as unary
    - much bigger than just semi-linear predicates

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
   - NL refers predicates decidable by log-space Turing machines, with input encoded as unary
   - much bigger than just semi-linear predicates
   - other extensions in the literature: clocks, cover-time service / absence-detection

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
   - NL refers predicates decidable by log-space Turing machines, with input encoded as unary
   - much bigger than just semi-linear predicates
   - other extensions in the literature: clocks, cover-time service / absence-detection
2. Study broadcasts in the computation-by-consensus paradigm

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
   - NL refers predicates decidable by log-space Turing machines, with input encoded as unary
   - much bigger than just semi-linear predicates
   - other extensions in the literature: clocks, cover-time service / absence-detection
2. Study broadcasts in the computation-by-consensus paradigm
3. Model global influences in e.g. biological systems (cf. [Bertrand et al. 2017])

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Why?

1. Simple extension to population protocols for NL power [Blondin, Esparza, Jaax 2019]
   - NL refers predicates decidable by log-space Turing machines, with input encoded as unary
   - much bigger than just semi-linear predicates
   - other extensions in the literature: clocks, cover-time service / absence-detection
2. Study broadcasts in the computation-by-consensus paradigm
3. Model global influences in e.g. biological systems (cf. [Bertrand et al. 2017])
4. Construct faster and more powerful protocols

Results

Prior work:

- Blondin, Esparza and Jaax show that BCPs compute exactly NL
  - no bounds on running time
  - multiple stages of reduction $\rightarrow$ complicated protocols

[1]w.r.t. number of transitions

Results

Prior work:

- Blondin, Esparza and Jaax show that BCPs compute exactly NL
  - no bounds on running time
  - multiple stages of reduction $\rightarrow$ complicated protocols

Our results:

1. time-optimal[1], simple protocols for semi-linear predicates
   - expected $\mathcal{O}(n \log n)$ transitions

[1] w.r.t. number of transitions

Results

Prior work:

- Blondin, Esparza and Jaax show that BCPs compute exactly NL
  - no bounds on running time
  - multiple stages of reduction $\rightarrow$ complicated protocols

Our results:

1. time-optimal[1], simple protocols for semi-linear predicates
   - expected $\mathcal{O}(n \log n)$ transitions
2. poly-time BCPs are precisely ZLP
   - i.e. predicates decidable by zero-error, log-space, expected poly-time randomised Turing Machines

[1] w.r.t. number of transitions

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Formally:

finite set of states Q,
transitions $B : Q \to Q \times Q^Q$

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

Formally:

$$\text{finite set of states } Q,$$
$$\text{transitions } B : Q \to Q \times Q^Q$$

- Pairwise interactions can be simulated

What is a Broadcast Consensus Protocol (BCP)?

$$\text{BCP} = \text{Population Protocol} + \text{Broadcasts}$$

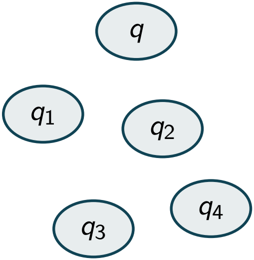
Formally:

$$\text{finite set of states } Q,$$
$$\text{transitions } B : Q \to Q \times Q^Q$$

- Pairwise interactions can be simulated
- Non-determinism can be simulated

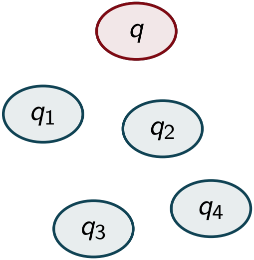
Transitions

Run on population of agents $C \in \mathbb{N}^Q$ (multiset of states).



Transitions

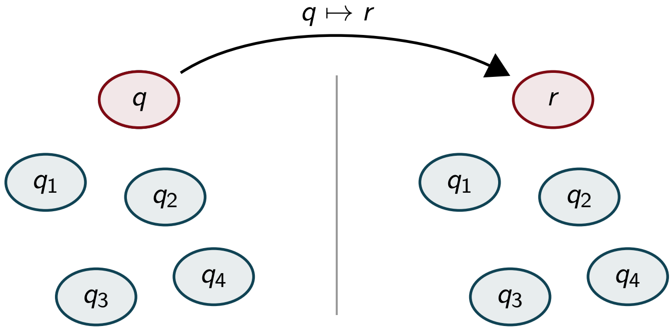
Run on population of agents $C \in \mathbb{N}^Q$ (multiset of states).



Execute: transition $q \mapsto r, f$, with $q, r \in Q, f : Q \to Q$

Transitions

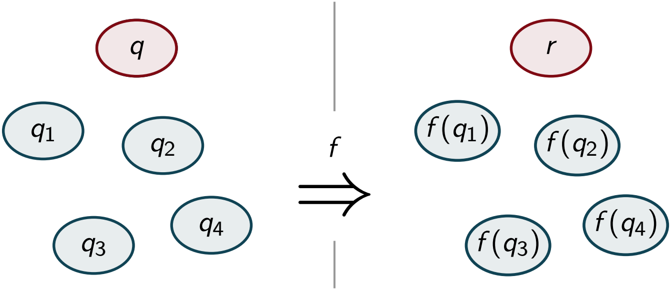
Run on population of agents $C \in \mathbb{N}^Q$ (multiset of states).



Execute: transition $q \mapsto r, f$, with $q, r \in Q, f : Q \to Q$

Transitions

Run on population of agents $C \in \mathbb{N}^Q$ (multiset of states).



Execute: transition $q \mapsto r, f$, with $q, r \in Q, f : Q \to Q$

Computation

initial states $I \subseteq Q$
output mapping $O : Q \to \{0, 1\}$
predicate $\varphi : \mathbb{N}^I \to \{0, 1\}$

Computation

initial states $I \subseteq Q$
output mapping $O : Q \to \{0, 1\}$
predicate $\varphi : \mathbb{N}^I \to \{0, 1\}$

How do we compute φ ?

Computation

initial states $I \subseteq Q$
output mapping $O : Q \to \{0, 1\}$
predicate $\varphi : \mathbb{N}^I \to \{0, 1\}$

How do we compute φ ?
Pick agents at random until everyone has (and retains) the same output.

Example

Majority $\varphi(x, y) \Leftrightarrow x \geq y$

$(x, y) = (2, 3)$

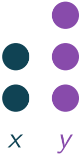
input

Example

Majority $\varphi(x, y) \Leftrightarrow x \geq y$

$(x, y) = (2, 3)$

input



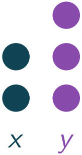
multiset

Example

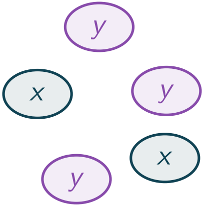
Majority $\varphi(x, y) \Leftrightarrow x \geq y$

$(x, y) = (2, 3)$

input

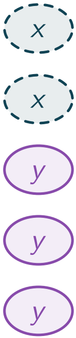


multiset



population

Compute $\varphi(x, y) \Leftrightarrow x \geq y$



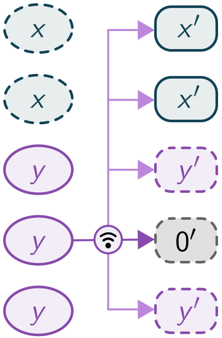
output 1

output 0

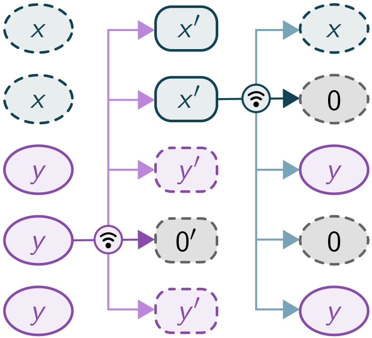
“active”

“disabled”

Compute $\varphi(x, y) \Leftrightarrow x \geq y$ $\qquad$ $y \mapsto 0, \{x \mapsto x', y \mapsto y', 0 \mapsto 0'\}$



Compute $\varphi(x, y) \Leftrightarrow x \geq y$ $\quad x' \mapsto 0, \{x' \mapsto x, y' \mapsto y, 0' \mapsto 0\}$



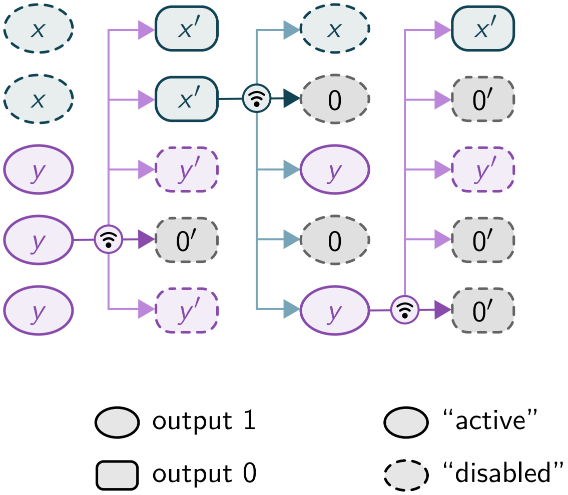
output 1

output 0

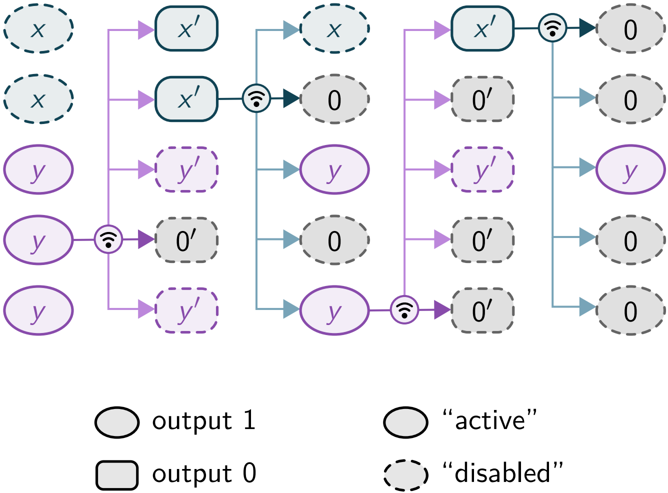
“active”

“disabled”

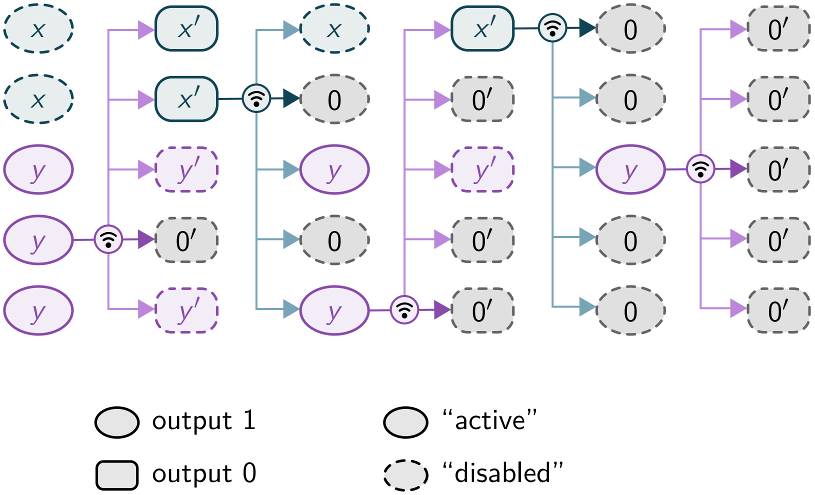
Compute $\varphi(x, y) \Leftrightarrow x \geq y$ $\quad$ $y \mapsto 0, \{x \mapsto x', y \mapsto y', 0 \mapsto 0'\}$



Compute $\varphi(x, y) \Leftrightarrow x \geq y$ $x' \mapsto 0, \{x' \mapsto x, y' \mapsto y, 0' \mapsto 0\}$



Compute $\varphi(x, y) \Leftrightarrow x \geq y$ $\quad\quad$ $y \mapsto 0, \{x \mapsto x', y \mapsto y', 0 \mapsto 0'\}$



Semi-linear predicates

Semi-linear predicates

- Example generalises to all semi-linear predicates

Semi-linear predicates

- Example generalises to all semi-linear predicates
- Shared global state

Semi-linear predicates

- Example generalises to all semi-linear predicates
- Shared global state

Steps:

1. Decompose semi-linear predicate into boolean combination of modulo and threshold predicates
2. Protocol for modulo predicates
3. Protocol for threshold predicates
4. Boolean combinations (simple)

Modulo predicates

$$a_1 x_1 + ... + a_l x_l \equiv b \pmod{k}$$

Global state is $\{0, ..., k-1\}$, additions modulo k

Threshold predicates

$$a_1 x_1 + ... + a_l x_l \geq k$$

Global state is large enough counter, take care not to overflow.

- Standard coupon-collector analysis for $\mathcal{O}(n \log n)$ transitions

- Standard coupon-collector analysis for $\mathcal{O}(n \log n)$ transitions
- Simple matching lower bound (all agents have to act at least once)

- Standard coupon-collector analysis for $\mathcal{O}(n \log n)$ transitions
- Simple matching lower bound (all agents have to act at least once)

Thus we get time-optimal BCPs for semi-linear predicates.

Thank you for
your attention!