Unreliable population protocols

Black ninjas in the dark, and they are not even there
Small birds fly away mid-conversation

Michael Raskin

Dept. of CS, TUM

February 27, 2019

What am I going to tell you

Population protocols
Related models
Expressive power

What happens with unreliable communication?
Rather general notion of unreliable protocol
...And general upper bounds

What am I going to tell you

Population protocols
Related models
Expressive power

What happens with unreliable communication?
Rather general notion of unreliable protocol
...And general upper bounds

Population protocols

Indistinguishable agents with constant-sized local storage are chosen nondeterministically to interact and update their state based on interaction.

... or maybe they send and receive messages instead (queued transmission protocols)
... or maybe they *broadcast* messages (broadcast protocols)

Small low-power sensors
... or simple coordination of large powerful servers
with fewer interesting corner cases

Population protocols

Indistinguishable agents with constant-sized local storage are chosen nondeterministically to interact and update their state based on interaction.

... or maybe they send and receive messages instead (queued transmission protocols)
... or maybe they *broadcast* messages (broadcast protocols)

Small low-power sensors
... or simple coordination of large powerful servers
with fewer interesting corner cases

Population protocols

Indistinguishable agents with constant-sized local storage are chosen nondeterministically to interact and update their state based on interaction.

... or maybe they send and receive messages instead (queued transmission protocols)
... or maybe they *broadcast* messages (broadcast protocols)

Small low-power sensors
... or simple coordination of large powerful servers
with fewer interesting corner cases

Population protocols: definition

A finite number of agents (doesn't change during execution)
A finite set Q of states
Each step: two agents are selected at random, communicate, change their own state accordingly
Transition relations $\rightarrow \subseteq Q^2 \times Q^2$

Input states $I \subset Q$
Each state has opinion, yes or no. Consensus is stable if no further execution can reach a non-consensus state (or consensus with different answer)
Protocol computes a predicate on $\mathbb{N}^I$ if a population described by an input always* converges to stable consensus on correct answer

* for some value of «always» — namely, in all fair runs: if some configuration can be reached, it will be reached

Population protocols: definition

A finite number of agents (doesn't change during execution)
A finite set Q of states
Each step: two agents are selected at random, communicate, change their own state accordingly
Transition relations $\rightarrow \subseteq Q^2 \times Q^2$

Input states $I \subset Q$
Each state has opinion, yes or no. Consensus is stable if no further execution can reach a non-consensus state (or consensus with different answer)
Protocol computes a predicate on $\mathbb{N}^I$ if a population described by an input always* converges to stable consensus on correct answer

* for some value of «always» — namely, in all fair runs: if some configuration can be reached, it will be reached

Population protocols: definition

A finite number of agents (doesn't change during execution)
A finite set Q of states
Each step: two agents are selected at random, communicate, change their own state accordingly
Transition relations $\rightarrow \subseteq Q^2 \times Q^2$

Input states $I \subset Q$
Each state has opinion, yes or no. Consensus is stable if no further execution can reach a non-consensus state (or consensus with different answer)
Protocol computes a predicate on $\mathbb{N}^I$ if a population described by an input always* converges to stable consensus on correct answer

* for some value of «always» — namely, in all fair runs: if some configuration can be reached, it will be reached

Population protocols: definition

A finite number of agents (doesn't change during execution)
A finite set Q of states
Each step: two agents are selected at random, communicate, change their own state accordingly
Transition relations $\rightarrow \subseteq Q^2 \times Q^2$

Input states $I \subset Q$
Each state has opinion, yes or no. Consensus is stable if no further execution can reach a non-consensus state (or consensus with different answer)
Protocol computes a predicate on $\mathbb{N}^I$ if a population described by an input always* converges to stable consensus on correct answer

* for some value of «always» — namely, in all fair runs: if some configuration can be reached, it will be reached

Population protocols: example

«Are there at least 5 agents?»

States: 0, 1, 2, 3, 4, 5
Input state: 1
Output: 5 means «yes»

$(m, n) \to (min(m + n, 5), 0)$
But $(5, k) \to (5, 5)$

Population protocols: example

«Are there at least 5 agents?»

States: 0, 1, 2, 3, 4, 5
Input state: 1
Output: 5 means «yes»

$(m, n) \to (min(m + n, 5), 0)$
But $(5, k) \to (5, 5)$

Population protocols: example

«Are there at least 5 agents?»

States: 0, 1, 2, 3, 4, 5
Input state: 1
Output: 5 means «yes»

$(m, n) \to (min(m + n, 5), 0)$
But $(5, k) \to (5, 5)$

Variants

- Broadcast — one transmits, updates on that; everyone else receive
- Immediate transmission — one sees the state, the other only presence
- Immediate observation — one observes, the other doesn't notice
- Delayed transmission — messages are sent, received at some later time
- Delayed observation — same, but sender can't change state
- Queued transmission — delayed transmission, but can refuse to consume message (keeping state)

Variants

- Broadcast — one transmits, updates on that; everyone else receive
- Immediate transmission — one sees the state, the other only presence
- Immediate observation — one observes, the other doesn't notice
- Delayed transmission — messages are sent, received at some later time
- Delayed observation — same, but sender can't change state
- Queued transmission — delayed transmission, but can refuse to consume message (keeping state)

Variants

- Broadcast — one transmits, updates on that; everyone else receive
- Immediate transmission — one sees the state, the other only presence
- Immediate observation — one observes, the other doesn't notice
- Delayed transmission — messages are sent, received at some later time
- Delayed observation — same, but sender can't change state
- Queued transmission — delayed transmission, but can refuse to consume message (keeping state)

Variants

- Broadcast — one transmits, updates on that; everyone else receive
- Immediate transmission — one sees the state, the other only presence
- Immediate observation — one observes, the other doesn't notice
- Delayed transmission — messages are sent, received at some later time
- Delayed observation — same, but sender can't change state
- Queued transmission — delayed transmission, but can refuse to consume message (keeping state)

Messages never lost,
two-party communication always transactional

Expressive power

Broadcast protocols — NSpace(n)

Population protocols, Queued transmission — semilinear sets (unions of multidimensional arithmetic progressions)

Immediate observation — counting sets («$x_j > const$» and combinations)
Delayed observation: 1-counting sets («$x_j > 0$» and combinations)

Immediate transmission, Delayed transmission — $SLIN \cap coreMOD$
expressible via $x_j \mod const = const$ for either-zero-or-large inputs;
$x > y \notin coreMOD$ but $x > y \wedge z = 1 \in coreMOD$
intermediate **between semilinear and counting**

Expressive power

Broadcast protocols — NSpace(n)

Population protocols, Queued transmission — semilinear sets (unions of multidimensional arithmetic progressions)

Immediate observation — counting sets («$x_j > const$» and combinations)
Delayed observation: 1-counting sets («$x_j > 0$» and combinations)

Immediate transmission, Delayed transmission — $SLIN \cap coreMOD$
expressible via $x_j \mod const = const$ for either-zero-or-large inputs;
$x > y \notin coreMOD$ but $x > y \wedge z = 1 \in coreMOD$
intermediate **between semilinear and counting**

Expressive power

Broadcast protocols — NSpace(n)

Population protocols, Queued transmission — semilinear sets (unions of multidimensional arithmetic progressions)

Immediate observation — counting sets («$x_j > const$» and combinations)
Delayed observation: 1-counting sets («$x_j > 0$» and combinations)

Immediate transmission, Delayed transmission — $SLIN \cap coreMOD$
expressible via $x_j \mod const = const$ for either-zero-or-large inputs;
$x > y \notin coreMOD$ but $x > y \wedge z = 1 \in coreMOD$
intermediate **between semilinear and counting**

Expressive power

Broadcast protocols — NSpace(n)

Population protocols, Queued transmission — semilinear sets (unions of multidimensional arithmetic progressions)

Immediate observation — counting sets («$x_j > const$» and combinations)
Delayed observation: 1-counting sets («$x_j > 0$» and combinations)

Immediate transmission, Delayed transmission — $SLIN \cap coreMOD$
expressible via $x_j \mod const = const$ for either-zero-or-large inputs;
$x > y \notin coreMOD$ but $x > y \wedge z = 1 \in coreMOD$
intermediate **between semilinear and counting**

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

$\mathbf{1}, \mathbf{1}, 1, 1$

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

$\mathbf{1}, 2, \mathbf{1}, 1$

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

2, **2**, 1, 1

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

$2, 3, \mathbf{1}, \mathbf{1}$

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

$2, \mathbf{3}, \mathbf{1}, 2$

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

2, 3, **3**, 2

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

3, **3**, 3, **2**

Immediate observation: $x \geq c$

Simplest case: compare total number of agents and c

$(k, k) \to (k+1, k)$
$(k, c) \to (c, c)$

3, 3, 3, 3

Generalisation

Agents, states, maybe message packets

Transition relation on configurations

Technical restriction: only agent states matter, adding extra agents or packets always possible

Fairness predicate on runs

Every finite run has fair continuation; doing nothing forever when something is possible to do is unfair

Generalisation

Agents, states, maybe message packets

Transition relation on configurations

Technical restriction: only agent states matter, adding extra agents or packets always possible

Fairness predicate on runs

Every finite run has fair continuation; doing nothing forever when something is possible to do is unfair

Generalisation

Agents, states, maybe message packets

Transition relation on configurations

Technical restriction: only agent states matter, adding extra agents or packets always possible

Fairness predicate on runs

Every finite run has fair continuation; doing nothing forever when something is possible to do is unfair

Unreliability

Agents, states, maybe message packets, transition relation...

We can always do «the same but some agents fail to update»

We keep track of activity — roughly, if broadcast is received, the sender noticed sending

Unreliability

Agents, states, maybe message packets, transition relation...

We can always do «the same but some agents fail to update»

We keep track of activity — roughly, if broadcast is received, the sender noticed sending

Unreliability

Agents, states, maybe message packets, transition relation...

We can always do «the same but some agents fail to update»

We keep track of activity — roughly, if broadcast is received, the sender noticed sending

Unreliability: expressive power

Immediate observation — counting sets
(nothing changes, and fairness condition doesn't matter)

This is the universal upper bound
(And verification tasks for IO population protocols are in PSPACE)

Message-based unreliable protocols cannot distinguish $\odot^1$ and $\odot^2$
Loss of expressive power is non-monotonic!
(Reliable Queued transmission is more powerful than Immediate observation)

Unreliability: expressive power

Immediate observation — counting sets
(nothing changes, and fairness condition doesn't matter)

This is the universal upper bound
(And verification tasks for IO population protocols are in PSPACE)

Message-based unreliable protocols cannot distinguish $\odot^1$ and $\odot^2$
Loss of expressive power is non-monotonic!
(Reliable Queued transmission is more powerful than Immediate observation)

Unreliability: expressive power

Immediate observation — counting sets
(nothing changes, and fairness condition doesn't matter)

This is the universal upper bound
(And verification tasks for IO population protocols are in PSPACE)

Message-based unreliable protocols cannot distinguish $\odot^1$ and $\odot^2$
Loss of expressive power is non-monotonic!
(Reliable Queued transmission is more powerful than Immediate observation)

Unreliability: expressive power

Immediate observation — counting sets
(nothing changes, and fairness condition doesn't matter)

This is the universal upper bound
(And verification tasks for IO population protocols are in PSPACE)

Message-based unreliable protocols cannot distinguish $\odot^1$ and $\odot^2$
Loss of expressive power is non-monotonic!
(Reliable Queued transmission is more powerful than Immediate observation)

Unreliability: expressive power

Immediate observation — counting sets
(nothing changes, and fairness condition doesn't matter)

This is the universal upper bound
(And verification tasks for IO population protocols are in PSPACE)

Message-based unreliable protocols cannot distinguish $\odot^1$ and $\odot^2$
Loss of expressive power is non-monotonic!
(Reliable Queued transmission is more powerful than Immediate observation)

Why? — General upper bound

One more shadow agent (copycat) lemma

Have my message been lost — or have someone taken my place?

Alternative histories: shadow agent starts from the same state, and can be removed without breaking execution correctness
Shadow agent could have any state its visible counterpart visited — and maybe more!
(does not work without unreliability —
$(m, n) \to (m + n, 0)$ preserves sum of the states)

... And extra agents in a very popular state do not matter

Why? — General upper bound

One more shadow agent (copycat) lemma

Have my message been lost — or have someone taken my place?

Alternative histories: shadow agent starts from the same state, and can be removed without breaking execution correctness
Shadow agent could have any state its visible counterpart visited — and maybe more!
(does not work without unreliability —
$(m, n) \to (m + n, 0)$ preserves sum of the states)

... And extra agents in a very popular state do not matter

Why? — General upper bound

One more shadow agent (copycat) lemma

Have my message been lost — or have someone taken my place?

Alternative histories: shadow agent starts from the same state, and can be removed without breaking execution correctness
Shadow agent could have any state its visible counterpart visited — and maybe more!
(does not work without unreliability —
$(m, n) \to (m + n, 0)$ preserves sum of the states)

... And extra agents in a very popular state do not matter

Why? — General upper bound

One more shadow agent (copycat) lemma

Have my message been lost — or have someone taken my place?

Alternative histories: shadow agent starts from the same state, and can be removed without breaking execution correctness
Shadow agent could have any state its visible counterpart visited — and maybe more!
(does not work without unreliability —
$(m, n) \to (m + n, 0)$ preserves sum of the states)

... And extra agents in a very popular state do not matter

Why? — Message-based unreliable protocols

Supply of messages

Do not consume rarities

Get blocked or build up supply of some messages

Burn rarities without updating state

Impossible to replace

Doesn't matter how many agents now

Why? — Message-based unreliable protocols

Supply of messages

Do not consume rarities

Get blocked or build up supply of some messages

Burn rarities without updating state

Impossible to replace

Doesn't matter how many agents now

Why? — Message-based unreliable protocols

Supply of messages
Do not consume rarities
Get blocked or build up supply of some messages
Burn rarities without updating state
Impossible to replace
Doesn't matter how many agents now

Why? — Message-based unreliable protocols

Supply of messages
Do not consume rarities
Get blocked or build up supply of some messages
Burn rarities without updating state
Impossible to replace
Doesn't matter how many agents now

Why? — Message-based unreliable protocols

Supply of messages
Do not consume rarities
Get blocked or build up supply of some messages
Burn rarities without updating state
Impossible to replace
Doesn't matter how many agents now

Conclusion

Immediate observation population protocol are nice!
Even when communication is unreliable

Non-monotonic impact of model limitations

Thanks for your attention

Questions?

(before the details)

Fairness conditions

Recurrent configuration: reached infinitely many times
Default fairness condition: if something is always reachable, it is recurrent

General conditions:
Eventual: can fairly continue everything finite
Activity-ensuring: only one recurrent configuration, then no possible step

Default condition satisfies both

Two general conditions enough to reach Immediate observation expressive power

Fairness conditions

Recurrent configuration: reached infinitely many times
Default fairness condition: if something is always reachable, it is recurrent

General conditions:
Eventual: can fairly continue everything finite
Activity-ensuring: only one recurrent configuration, then no possible step

Default condition satisfies both

Two general conditions enough to reach Immediate observation expressive power

Fairness conditions

Recurrent configuration: reached infinitely many times
Default fairness condition: if something is always reachable, it is recurrent

General conditions:
Eventual: can fairly continue everything finite
Activity-ensuring: only one recurrent configuration, then no possible step

Default condition satisfies both

Two general conditions enough to reach Immediate observation expressive power

Fairness conditions

Recurrent configuration: reached infinitely many times
Default fairness condition: if something is always reachable, it is recurrent

General conditions:
Eventual: can fairly continue everything finite
Activity-ensuring: only one recurrent configuration, then no possible step

Default condition satisfies both

Two general conditions enough to reach Immediate observation expressive power

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

No messages:
Random run almost surely fair
Recurrent configuration set of fair run —
positive probability of being recurrent configuration set randomly

Proof:
Finite reachable configuration set — lower bound for every multistep transition probability
Reaching member of recurrent set — having almost surely the same recurrent set

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

No messages:
Random run almost surely fair
Recurrent configuration set of fair run —
positive probability of being recurrent configuration set randomly

Proof:
Finite reachable configuration set — lower bound for every multistep transition probability
Reaching member of recurrent set — having almost surely the same recurrent set

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

No messages:
Random run almost surely fair
Recurrent configuration set of fair run —
positive probability of being recurrent configuration set randomly

Proof:
Finite reachable configuration set — lower bound for every multistep transition probability
Reaching member of recurrent set — having almost surely the same recurrent set

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

Messages:
Activity condition has probability 1
Every condition with probability 1 is eventual

Proof:
Not taking positive-probability step forever — probability 0
Any non-eventual condition fails with positive probability

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

Messages:
Activity condition has probability 1
Every condition with probability 1 is eventual

Proof:
Not taking positive-probability step forever — probability 0
Any non-eventual condition fails with positive probability

Fairness conditions: probabilistic

Positive probabilities for possible steps
Steps are independent

Messages:
Activity condition has probability 1
Every condition with probability 1 is eventual

Proof:
Not taking positive-probability step forever — probability 0
Any non-eventual condition fails with positive probability

Unreliable protocol: possible failures

«It's loud here, did we even hear each other?»

State changes can fail
Defined active agents (transmitters)
If passive agents update state, **all** active agents do

Shadow agents lemma

Shadow agent: can be removed, still a legal execution
Alternative histories: each agent has shadow agents — same initial state (one per history)

Never enter state without one of your shadows already there

Try fair continuation; for steps without shadow in place — replace by ambiguity «failed update/shadow stealing place», try new fair continuation
Finite number of restarts — eventually, shadows everywhere

Shadow agents lemma

Shadow agent: can be removed, still a legal execution
Alternative histories: each agent has shadow agents — same initial state (one per history)

Never enter state without one of your shadows already there

Try fair continuation; for steps without shadow in place — replace by ambiguity «failed update/shadow stealing place», try new fair continuation
Finite number of restarts — eventually, shadows everywhere

Truncatable protocols

Lack of stable consensus — upward closed
Dickson's lemma: minimal non-stable-consensus configurations (finite set)

Stable consensus doesn't care about exact counts above all of these

Popular input state has many agents go to popular state in consensus
Shadow agents lemma: popular input state can get extra agent with same target state

This yields the same consensus value, so adding agents to popular input states doesn't matter

Truncatable protocols

Lack of stable consensus — upward closed
Dickson's lemma: minimal non-stable-consensus configurations (finite set)

Stable consensus doesn't care about exact counts above all of these

Popular input state has many agents go to popular state in consensus
Shadow agents lemma: popular input state can get extra agent with same target state

This yields the same consensus value, so adding agents to popular input states doesn't matter

Truncatable protocols

Lack of stable consensus — upward closed
Dickson's lemma: minimal non-stable-consensus configurations (finite set)

Stable consensus doesn't care about exact counts above all of these

Popular input state has many agents go to popular state in consensus
Shadow agents lemma: popular input state can get extra agent with same target state

This yields the same consensus value, so adding agents to popular input states doesn't matter

Message-based unreliable protocols

Only one agent involved in step — either sending or receiving

Cannot distinguish one agent from two

Proof idea:
Define «lots» of copies for messages
The more types are abundant, the lower «lots»
Forbid benefitting from rare messages

Creation of rare messages has to make some message abundant

When rare messages cannot be created, kill existing ones (fail to update)

Repeat the same with two agents in parallel

Reachable states are the same — so consensus is the same

Message-based unreliable protocols

Only one agent involved in step — either sending or receiving

Cannot distinguish one agent from two

Proof idea:
Define «lots» of copies for messages
The more types are abundant, the lower «lots»
Forbid benefitting from rare messages

Creation of rare messages has to make some message abundant

When rare messages cannot be created, kill existing ones (fail to update)

Repeat the same with two agents in parallel

Reachable states are the same — so consensus is the same

Message-based unreliable protocols

Only one agent involved in step — either sending or receiving

Cannot distinguish one agent from two

Proof idea:
Define «lots» of copies for messages
The more types are abundant, the lower «lots»
Forbid benefitting from rare messages

Creation of rare messages has to make some message abundant

When rare messages cannot be created, kill existing ones (fail to update)

Repeat the same with two agents in parallel

Reachable states are the same — so consensus is the same

Message-based unreliable protocols

Only one agent involved in step — either sending or receiving

Cannot distinguish one agent from two

Proof idea:
Define «lots» of copies for messages
The more types are abundant, the lower «lots»
Forbid benefitting from rare messages

Creation of rare messages has to make some message abundant

When rare messages cannot be created, kill existing ones (fail to update)

Repeat the same with two agents in parallel

Reachable states are the same — so consensus is the same

Message-based unreliable protocols

Only one agent involved in step — either sending or receiving

Cannot distinguish one agent from two

Proof idea:
Define «lots» of copies for messages
The more types are abundant, the lower «lots»
Forbid benefitting from rare messages

Creation of rare messages has to make some message abundant

When rare messages cannot be created, kill existing ones (fail to update)

Repeat the same with two agents in parallel

Reachable states are the same — so consensus is the same

Immediate observation protocols

Natively unreliability-tolerant

Most expressive among unreliable

Merely PSPACE verification problems

Use them!

Immediate observation protocols

Natively unreliability-tolerant

Most expressive among unreliable

Merely PSPACE verification problems

Simply cool

Use them!

Thanks for your attention

Questions?

(last call!)