Modular population protocols

Michael Raskin

LaBRI, University of Bordeaux, CNRS UMR 5800

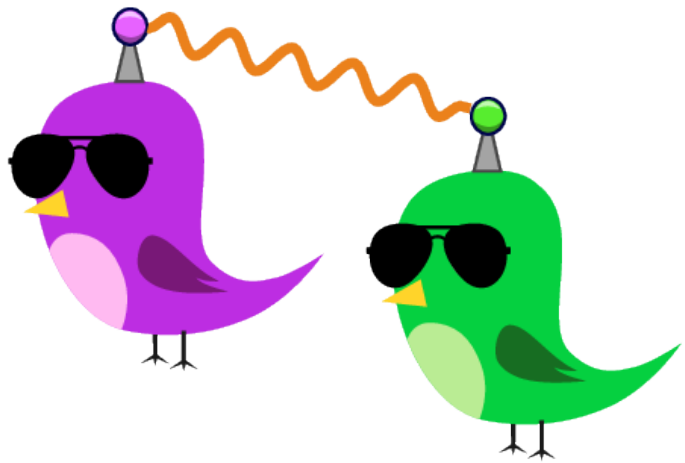
2024-09-05

Population protocols — original

Small sensors: constant memory, identical,
pairwise interaction when accidentally near

Accepting and rejecting states
Aiming for eventual consensus — unanimous acceptance or rejection
(of initial configuration)

[AADFP 2004]

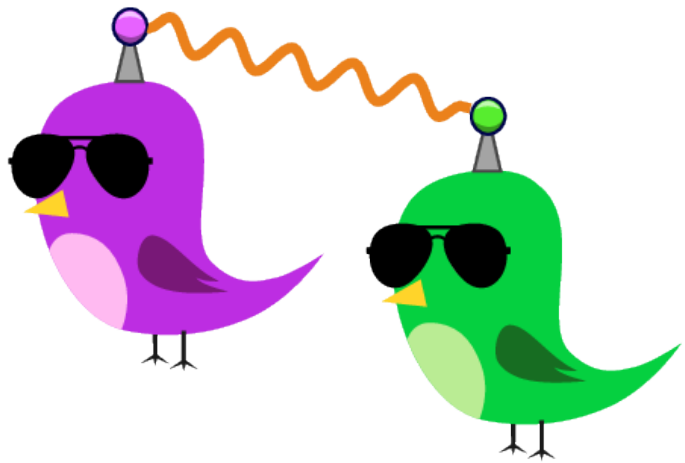


Population protocols — original

Small sensors: constant memory, identical,
pairwise interaction when accidentally near

Accepting and rejecting states
Aiming for eventual consensus — unanimous acceptance or rejection
(of initial configuration)

[AADFP 2004]

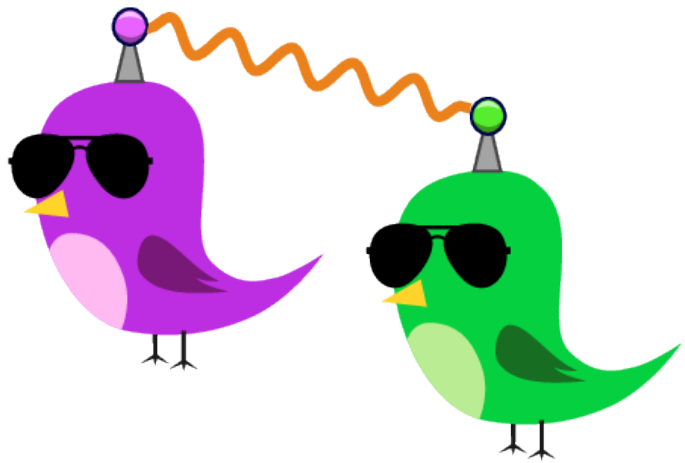


Population protocols — original

Small sensors: constant memory, identical,
pairwise interaction when accidentally near

Accepting and rejecting states
Aiming for eventual consensus — unanimous acceptance or rejection
(of initial configuration)

[AADFP 2004]



Population protocols — applicability

Original motivation — small sensors, too small to be plausible today

Edge of decidability
Even beefy servers like simplicity!

Population protocols — applicability

Original motivation — small sensors, too small to be plausible today

Edge of decidability
Even beefy servers like simplicity!

Population protocols — tasks for protocols

Eventual consensus on a simple predicate over unchanging inputs — limited ambitions

Also, most modular constructions are completely ad-hoc

Aims:

- Modularity
- Generality of tasks

Population protocols — tasks for protocols

Eventual consensus on a simple predicate over unchanging inputs — limited ambitions

Also, most modular constructions are completely ad-hoc

Aims:

- Modularity
- Generality of tasks

Population protocols — tasks for protocols

Eventual consensus on a simple predicate over unchanging inputs — limited ambitions

Also, most modular constructions are completely ad-hoc

Aims:

- Modularity
- Generality of tasks

Model problem

Role distribution in a distributed system

- Servers have different configurations
- Servers need to be assigned roles
- Roles have configuration requirements
- Balance of servers assigned to different roles

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

Model problem

Role distribution in a distributed system

- Servers have different configurations
- Servers need to be assigned roles
- Roles have configuration requirements
- Balance of servers assigned to different roles

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

Model problem

Role distribution in a distributed system

- Servers have different configurations
- Servers need to be assigned roles
- Roles have configuration requirements
- Balance of servers assigned to different roles

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

Model problem

Role distribution in a distributed system

- Servers have different configurations
- Servers need to be assigned roles
- Roles have configuration requirements
- Balance of servers assigned to different roles

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

Model problem

Role distribution in a distributed system

- Servers have different configurations
- Servers need to be assigned roles
- Roles have configuration requirements
- Balance of servers assigned to different roles

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

Model problem — evolving situation

«Storage servers need more storage than web front-ends, but more storage on web front-ends doesn't hurt, and we need the same number of the two types whenever possible»

- Can schedule server for shutdown and maintenance/removal
- Can add servers
- Can change server configuration

Execution model

Agents have inputs they can't change, and internal states
Step options:

- Normal protocol step
- Reconfiguration

Reconfigurations:
Special input $\perp$ means request to shutdown,
special state $\perp$ means the agent did shut down

- Add an agent that is shut down and has shutdown input
- Remove an agent that is shut down and has shutdown input
- Change an agent's input

Execution model

Agents have inputs they can't change, and internal states
Step options:

- Normal protocol step
- Reconfiguration

Reconfigurations:
Special input $\perp$ means request to shutdown,
special state $\perp$ means the agent did shut down

- Add an agent that is shut down and has shutdown input
- Remove an agent that is shut down and has shutdown input
- Change an agent's input

Specifications

Two parts to specify

- Single-agent (local) input-output compatibility
  («32 GB storage not enough for storage servers»)
- Multiset-of-inputs to multiset-of-outputs (global) compatibility
  («Most of the servers have enough storage?
  Then webservers-to-storage-servers should be an equal split»)

After reconfigurations cease, each fair execution should eventually:

- Satisfy shutdown requests
- Stabilise all agent outputs
- Satisfy both parts of the specification

Specifications

Two parts to specify

- Single-agent (local) input-output compatibility
  («32 GB storage not enough for storage servers»)
- Multiset-of-inputs to multiset-of-outputs (global) compatibility
  («Most of the servers have enough storage?
  Then webservers-to-storage-servers should be an equal split»)

After reconfigurations cease, each fair execution should eventually:

- Satisfy shutdown requests
- Stabilise all agent outputs
- Satisfy both parts of the specification

Specifications

Two parts to specify

- Single-agent (local) input-output compatibility
  («32 GB storage not enough for storage servers»)
- Multiset-of-inputs to multiset-of-outputs (global) compatibility
  («Most of the servers have enough storage?
  Then webservers-to-storage-servers should be an equal split»)

After reconfigurations cease, each fair execution should eventually:

- Satisfy shutdown requests
- Stabilise all agent outputs
- Satisfy both parts of the specification

Composition

Given two protocols implementing two specifications,
can compose them:

- Each agent keeps the states for both protocols
- Input is used by the first protocol
- Output of the first protocol is used as the input for the second
- Output of the second protocol is used as the output of the composition

This implements composition of specifications:
global and local compatibility relations are composed as multi-valued functions

Composition

Given two protocols implementing two specifications,
can compose them:

- Each agent keeps the states for both protocols
- Input is used by the first protocol
- Output of the first protocol is used as the input for the second
- Output of the second protocol is used as the output of the composition

This implements composition of specifications:
global and local compatibility relations are composed as multi-valued functions

Expressive power

A specification can be unimplementable for specific multisets of inputs

«Exact same number of web and storage servers, even if the total is odd»

Otherwise, if global compatibility is semilinear, the specification can be implemented by population protocols with reconfiguration

A specification may be implementable for wrong reasons
«Only web front-ends, or a prime number of storage servers»

For any specification implemented by a protocol, another specification exists:

- Implemented by another protocol
- More restrictive (forbids everything forbidden by the original)
- With semilinear global compatibility

Expressive power

A specification can be unimplementable for specific multisets of inputs

«Exact same number of web and storage servers, even if the total is odd»

Otherwise, if global compatibility is semilinear, the specification can be implemented by population protocols with reconfiguration

A specification may be implementable for wrong reasons
«Only web front-ends, or a prime number of storage servers»

For any specification implemented by a protocol, another specification exists:

- Implemented by another protocol
- More restrictive (forbids everything forbidden by the original)
- With semilinear global compatibility

Expressive power

A specification can be unimplementable for specific multisets of inputs

«Exact same number of web and storage servers, even if the total is odd»

Otherwise, if global compatibility is semilinear, the specification can be implemented by population protocols with reconfiguration

A specification may be implementable for wrong reasons
«Only web front-ends, or a prime number of storage servers»

For any specification implemented by a protocol, another specification exists:

- Implemented by another protocol
- More restrictive (forbids everything forbidden by the original)
- With semilinear global compatibility

Expressive power

A specification can be unimplementable for specific multisets of inputs

«Exact same number of web and storage servers, even if the total is odd»

Otherwise, if global compatibility is semilinear, the specification can be implemented by population protocols with reconfiguration

A specification may be implementable for wrong reasons
«Only web front-ends, or a prime number of storage servers»

For any specification implemented by a protocol, another specification exists:

- Implemented by another protocol
- More restrictive (forbids everything forbidden by the original)
- With semilinear global compatibility

Conclusion

- A larger class of «tasks» can be defined for population protocols
- Cover more of the interesting tasks...
- ... and gain modularity
- Expressive power is still «semilinear»

- Future work: redefine and generalize known «efficient» protocols in this form

Thanks for your attention

Questions?

Conclusion

- A larger class of «tasks» can be defined for population protocols
- Cover more of the interesting tasks...
- ... and gain modularity
- Expressive power is still «semilinear»

- Future work: redefine and generalize known «efficient» protocols in this form