Fairness in Population Protocols

Ioannis Chatzigiannakis
Othon Michail
Paul Spirakis
Shlomi Dolev
Sandor Fekete

September 2009
Dugstuhl 09371

Networked Societies of Tiny Artefacts

- Our life is now full of small devices
  - communicate with each other when they are close.
- Such devices form networks
  - potentially support myriads of new and exciting applications.
- Technology would like such systems to be dependable and adaptive
  - to the user needs,
  - sudden changes of the environment,
  - specific applications characteristics.

Modelling Assumptions

- Each device is severely limited
  - constant storage capacity (independent of n),
  - limited processing capabilities,
  - limited communication capabilities.
- Devices move passively (attached to mobile objects).
- Devices do not operate continuously.
- The designer cannot control
  - the motion of the devices,
  - how devices interact.

Significant Properties of Population Protocols (1)

- **Uniformity:** Protocol descriptions are independent of the population size n.
  - If a property holds for a small population size, it also holds for a large population size.
  - It suffices to verify a protocol in small population sizes.
- **Anonymity:** There is no room in the state of an agent to store a unique identifier.
  - A property cannot depend on the existence of specific devices.

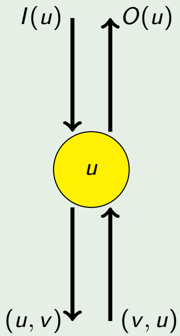
Significant Properties of Population Protocols (2)

- **Unpredictability:** The way in which agents interact is not controlled by the protocol.
    - The choice of which agents interact is made by an adversary.
    - A strong global fairness condition is imposed on the adversary to ensure the protocol makes progress.
- **Convergence:** Population protocols generally cannot detect when they have finished.
    - The agents' outputs are required to converge after some finite time to a common, correct value.

Interacting Automata

- Devices execute a software agent.
- Agents have constant memory.
- Initially receive a finite input.
- "Message exchanges" or "Shared Memory" are replaced by "Agent interactions".
    - One-way interactions.
    - Let u at state p, and v at state q, $(p, q) \mapsto (p', q')$
- Produce (eventually) an output.
- Do not halt.

Agent u



(Interactions with agent v)

Formal Definition of Population Protocols

[Angluin, Aspnes, Diamadi, Fischer, and Peralta, PODC '04]

Population V of $|V| = n$ agents.

A Population Protocol $\mathcal{A}$ consists of

- finite **input and output alphabets** X and Y,
- finite set of **states** Q,
- **input function** $I : X \to Q$,
- **output function** $O : Q \to Y$,
- **transition function** $\delta : Q \times Q \to Q \times Q$.

$\delta(p, q) = (p', q')$ or simply $(p, q) \mapsto (p', q')$ is called a **transition**.

Configurations, Executions

$C : V \rightarrow Q$, **population configuration** specifying the state of each agent

- $C \rightarrow C'$, C can go to C' in one step

Execution: Finite or infinite sequence $C_0, C_1, C_2, \ldots$, s.t. $C_i \rightarrow C_{i+1}$ for all i.

Scheduler & Computations

- The order in which pairs of agents interact is unpredictable.
  - We think of the schedule of interactions as being chosen by an adversary,
  - so that protocols must work correctly under any schedule the adversary may choose.
- We need to make the scheduler "*computation-friendly*" by a **fairness assumption**
  - Do not allow avoidance of a possible step forever.

Fairness Formally: For all C, C' s.t. $C \to C'$, if C occurs infinitely often in the execution the same holds for C'.

Computation: Infinite fair execution.

Problem Description

"*Find if at least* 5 *sensors have detected elevated temperature*."

- Each agent senses the temperature of a distinct bird after a global start signal.
- If detected elevated temperature input 1, else 0 (i.e. $X = \{0, 1\}$).
- We want every agent to eventually output
  - 1, if at least 5 birds were found sick,
  - 0, otherwise.

A Protocol

- $X = Y = \{0, 1\}$
- $Q = \{q_0, q_1, \ldots, q_5\}$,
- $I(0) = q_0$ and $I(1) = q_1$,
- $O(q_i) = 0$, for $0 \leq i \leq 4$, and $O(q_5) = 1$,
- δ:

$$\begin{aligned}(q_i, q_j) &\rightarrow (q_{i+j}, q_0), \text{ if } i + j < 5\\ &\rightarrow (q_5, q_5), \text{ otherwise.}\end{aligned}$$

Why it Works...

- Due to fairness, all agents with non-zero state index will eventually interact with each other.
- In each such interaction one of them keeps the sum.
- All indices are eventually aggregated in one agent's state index j (assuming that no faults can happen).
- If $j < 5$, then q_5 cannot occur, thus no agent ever outputs 1.
- Otherwise, state q_5 appears and floods the population (2nd rule), i.e. eventually every agent outputs 1.

Random interactions

- We wish to study the performance of a protocol
  - Total number of interactions to (stably) output
  - Average number of interactions per agent
- We replace the adversarial (but fair) scheduler with a more constrained interaction pattern
  - A Propabilistic Scheduler
  - For each configuration C defines an infinite sequence of probability distributions over all reachable configurations.
  - The scheduler needs to be Consistent
  - .. any time the scheduler encounters configuration C it chooses the next configuration with the same probability distribution.

Random Scheduler

[Angluin, Aspnes, Diamadi, Fischer, and Peralta: Distributed Computing, 18(4):235-253, 2006]

- The simplest probabilistic scheduler.
- Each pair of agents is equally likely to interact at each step.
- To generate C_{i+1}
  - selects an ordered pair $(u, v) \in E$ at random, independently and uniformly (each with probability $1/|E|$),
  - applies the transition function to $(C_i(u), C_i(v))$.
- The Random Scheduler is fair with probability 1.

State Scheduler – Motivation

- Consider a population protocol for k-mutual exclusion, in which only k agents are in state 1 and the rest of the population is in state 0.
  - When an agent that holds a token interacts with another agent, it passes the token.
  - Now consider an execution where $n \gg k$ and we use the Random Scheduler.
  - The probability of selecting a pair with states $(1, 0)$ is much smaller than selecting a pair with states $(0, 0)$.
  - ... the scheduler may initiate a large number of interactions that do not help the protocol in making progress.

State Scheduler

- The scheduler selects a pair based on the states of the processes.
  - In is aware of the Protocol structure.
- First selects a pair of *states* and in the sequel it selects one process from each state.
  - "meaningful" transitions to be selected more often
- The State Scheduler is fair with probability 1.

Transition Function Scheduler

- Continuing the same argument, we define one more scheduler.
  - Also assumes knowledge of the protocol executed.
- It examines the transition function δ
  - selects pairs of agents based on the defined transitions.
  - transitions that do not change the state, neither of the initiator nor of the responder agent (e.g., $(\alpha, \beta) \rightarrow (\alpha, \beta)$) are ignored.
  - Guarantees that all interactions will lead to a state change of either the initiator or the responder or both.
- The Transition Function Scheduler is fair with probability 1.

OR Protocol – One-way Epidemic Protocol

[Aspnes, Rupert: BEATCS, 93:98-117, 2007]

- Consider a simplified version of the example.
  - each agent with input 0 simply outputs 1 as soon as it interacts with some agent in state 1.
- $Q = X = Y = \{0, 1\}$
- The transitions defined by δ are the following:

$$\begin{array}{ll} (0,0) \rightarrow (0,0) & (1,0) \rightarrow (1,1) \\ (0,1) \rightarrow (1,1) & (1,1) \rightarrow (1,1) \end{array}$$

- Essentially, if all agents have input 0, no agent will ever be in state 1.
- If some agent has input 1, given a fair scheduler, we expect that the number of agents with state 1 will eventually reach n.
- "how fast is stability reached?"
- "how do different schedulers affect the performance of the protocol?"
- "are all (fair) schedulers equivalent?"

OR Protocol + Random Scheduler

[Angluin, Aspnes, Eisenstat: Distributed Computing, 21(3): 183-199, 2008]

- The number of interactions to complete is $\Theta(n \log n)$ w.h.p.
- Using arguments from the well-known coupon collector problem.

OR Protocol + State / Transition Function Scheduler

- The State Scheduler and the Transition Function Scheduler both require only $\mathcal{O}(n)$ interactions.
- The performance of a population protocol clearly depends on the scheduler's functionality.
- It seems that with additional knowledge, we can schedule interaction patterns that always lead to optimal computations.
- ... may also allow the definition of fair schedulers that lead the protocols to worst-case scenarios.

Modified Scheduler

- Selects a transition that leaves both the state of the initiator and that of the responder unaffected with probability $1 - \varepsilon$
- and from all the remaining transitions with probability ε, where $0 < \varepsilon < 1$.
- Those probabilities are then evenly divided into the corresponding transitions.
- The Modified Scheduler is fair with probability 1.

The Modified Scheduler can lead the OR Protocol to arbitrarily bad performance without violating the fairness of the scheduler..

Not all Fair Probabilistic Schedulers are Equivalent

- Not all fair probabilistic schedulers are time equivalent
- The fairness condition allows the definition of schedulers that may lead not only to optimal but also to worst-case running time scenarios.
- ... more surpisingly ...
- Not all fair probabilistic schedulers are computationally equivalent
- Protocols may be correct with some scheduler and err with others !

Majority Protocol – Two-way Epidemic Protocol

[Angluin, Aspnes, Eisenstat: DISC 2007]

- Assume that each agent initially votes
  - one of some election candidates x and y
  - or chooses to vote blank, denoted by b.
- If x is the majority vote, then we want every agent in the population to eventually output x, otherwise y
- The transitions defined by δ are the following:

$$\begin{array}{ll} (x, b) \rightarrow (x, x) & \quad (x, y) \rightarrow (x, b) \\ (y, b) \rightarrow (y, y) & \quad (y, x) \rightarrow (y, b) \end{array}$$

Majority Protocol + Random Scheduler
[Angluin, Aspnes, Eisenstat: DISC 2007]

- With high probability consensus is reached in $\mathcal{O}(n \log n)$ interactions
- The value chosen is the majority provided that its initial margin is $\omega(\sqrt{n \log n})$

Majority Protocol + Transition Function Scheduler

- This is not the case when the underlying scheduler is the Transition Function Scheduler.
- ... does not take into great account the advantage of xs

Lemma

The Majority Protocol errs under the Transition Function Scheduler with constant probability, when $x = \Theta(y)$ in the initial configuration.

Future Directions (1)

- We need to give a stronger definition of fairness,
- or impose further restrictions on the allowed schedulers.
- Population Protocols model may be useful for computer-aided verification
- Very important if we want to apply our protocols to real-critical application
- How can someone verify safely and quickly, in a distributed or centralized way, that a specific protocol meets its design objectives?

Future Directions (2)

- The computation power of Population protocols has been studied extensively.
  - A predicate is computable in the basic population protocol model if and only if it is semilinear.
- Can we give more power ?
- What if communication links equipped with a constant size buffer ?
  - Mediated Population Protocols
    Chatzigiannakis, Michail, Spirakis: ICALP 2009
  - Computationally stronger than the Population Protocol model.
  - Many new directions: finding subgraphs, deciding graph properties, optimization, approximation..
  - We need an exact characterization of the class of solvable problems
- What if Agents have unique identifiers ?
  - Community Protocols
    Guerraoui and Ruppert: ICALP 2009
  - Can tolerate byzantine failures.
  - Can solve any decision problem in $NSPACE(n \log n)$

- This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).
- FRONTS is a joint effort of eleven academic and research institutes in foundational algorithmic research in Europe.
- The effort is towards establishing the **foundations of adaptive networked societies of tiny artefacts**.

Thank You!