# Population Protocols for Adaptive MapReduce Applications
in Large-Scale Ad Hoc Networks of Tiny Mobile Artefacts *

[Experiment Abstract]

July 23, 2009

## 1 Introduction

### 1.1 Population Protocol

We use the Population Protocol model of Angluin et al. [2–4] in order to model large scale networks of very small resource-limited artefacts (agents) that are possibly mobile. Interaction happens between pairs of artefacts and according to a communication graph, as in Chatzigiannakis et al. [6–9]. These agents interact on a random bases; one pair of agents at a time. The goal of Population Protocols is to complete a task using these random interactions.

### 1.2 MapReduce processes

The studied population protocols facilitate MapReduce applications of Dean and Ghemawat [10]. The process of a MapReduce application is initiated by a distinguished agent, named the *initiator*. When the initiator interacts with an agent, it may take the *Map* step and assign a (part of the) task to that agent. Moreover, that agent may further map the task to another agent; in this case we say that this is an *intermediate* agent and if no further Map steps are taken we say that this is a *terminal* one. Terminal and intermediate agents that have completed their task, take the *Reduce* step. This step is taken upon interaction with intermediate agents until eventuality the interaction occurs with the initiator. Then, the initiator *shuts down* the MapReduce process after the task is completed.

### 1.3 Searching for a pair of primes

Let us consider an example in which the initiator is searching for pairs of primes, $p$ and $q$, such that $p, q \geq n_{min}$ and $pq \geq n_{max}$, where $n_{min}$ and $n_{max}$ are values defined by the initiator. (The cryptographical aspects of this problem are not within the scope of this experiment.) In order to do that, the initiator invokes a MapReduce process. The agents' interactions execute the MapReduce processes. On the Map step, the initiator (or an intermediate agent) defines the range of search, i.e., $n_{min}$ and $n_{max}$. The terminal agent, which takes the map step, randomly choose a candidate numbers within the rage $[n_{min}, n_{max}]$ and verifies primality using the tests of Miller-Rabin [19] or Agrawal-Kayal-Saxena [1]. In case that the primality test result is positive, the terminal agent delivers that number of an intermediate agent using the reduce step. The intermediate agent stores the tested prime number and continues mapping the task to additional terminal agents until a second prime number is found. When that occurs, the multiplication of these two primes is verified to be within the required bounds, i.e., $pq \geq n_{max}$. If the pair of primes is suitable, the intermediate agents take reduce steps until eventually the pair is delivered to the initiator. Once the task is completed, the limited resources of the tiny artefacts require that the initiator shuts down the MapReduce process.

## 2 Efficient implementation

The above example illustrates a difficulty in designing an efficient implementation of the Population Protocol model. On one hand, the system designer may wish to map the task to as may agents as possible. On the other hand, once the task is completed, the shutdown operation should be completed as soon as possible. The random interaction of agents may delay the execution of the shutdown operation. We propose the proposed experiment is to show that an efficient implementation indeed exists.

In the proposed experiment, we consider scenarios of mobile ad hoc networks (MANETs) that execute a MapReduce process for searching a pair of primes. In the proposed implementation, each mobile computer emulates a terminal agent that is requested to validate the primality of a random number, $n \in [n_{min}, n_{max}]$. After a number was tested, the terminal agent takes a Reduce step and delivers the tested result to the intermediate agent. We use Virtual Stationary Automata (VSAs) of Dolev et al. [11] for emulating intermediate agents. The VSAs are themselves emulated by the mobile computers. The VSAs are located at prespecified regions that tile the plane, defining a static virtual infrastructure [5]. Each mobile computer interacts with a single VSA; the one that the computer resides within its region.

We consider three possibilities for emulating the initiator agent.

1. The initiator is emulated by a particular VSA. It may happen that all the mobile computers leave the region of that VSA. When that happens, there is no initiator to receive the pair of primes and to shut down the MapReduce process.

2. The initiator is emulated by an autonomous virtual mobile node (AVMN) of Dolev et al. [12]. The AVMN can interact with VSAs that are within its area. Moreover, it can move to the more populated VSAs' areas, saying, by taking a greedy strategy that prefers moving into the most populated nearby areas. By taken such strategies, the lifespan of the AVMN increases. (We note that the AVMN emulates the initiator's first Map step, however, later Map steps can be taken intermediate agents and emulated by VSAs.)

3. One may consider an initiator agent that is emulated by a VSA. The VSA uses other VSAs in order to store the state of location of the initiator in a robust manner [see 14].

Another aspect of efficient implementation of the MapReduce process is the schedule of interaction among VSAs. We propose to use the scheme of Polygonal Broadcast by Dolev et al. [13].

## 3 Adaptivity and Robustness

Some MapReduce applications require the *Remap* functionally in which the intermediate agents reassign the same task but with different parameters. For example, one may consider a search for a large set of prime numbers rather than a pair of them. A MapReduce application that implements the Sieve algorithm of Eratosthenes (or modern versions of it, such as [16–18]) benefits from mapping the search assignments together with a set of known prime numbers. In this case, remapping the task refer to actions that delivers to the terminal agents new sets of prime numbers that were discovered by other terminal agents.

In order to implement Remap operations, the MapReduce processes must adapt their internal structure during the process execution. Moreover, they must adapt their external structure in order to deal with mobility. Namely, the Polygonal Broadcast by Dolev et al. [13] has to be replaced with an adaptive internal structure. For example, the internal structure of the system can be maintained by using an adaptive and self-stabilizing forest of Dolev et al. [15]; the tree allows to keep the context computation that is required for the correct remapping of tasks. We note that the robustness of this approach depends on evolvement of the communication graph and the rate in which topological changes occur.

We propose to experiment with an adaptive structure that efficiently uses redundancy and does not depend on the rate in which topological changes occur. In consider an implementation of the Population Protocol model in which there are *harbor* intermediate agents. The harbor agents are a fix set that maintain the internal and external structures. They perform Map, Remap and Reduce operation while maintaining the task's context, i.e., bookeeping of the MapReduce operations.

We consider harbors that are emulated by VSAs. We propose to overcome the possibility of harbor dissapearance/mulfunctuions by allowing harbors to coordinate their actions and creating replicas for the sack of redundancy (e.g., primary-back schemes). The coordination among these harbors may use reliable *ships* that are emulated by AVMAs [12]. Another way it to let the harbors register their state at their home locations [14].

## 4 Existing code

We have implemented VSA and AVMN using JiST/SWANS (so we are able to consider different mobility models, including real-life VANET). There are several open sources libraries for testing primality, e.g., `http://www.gridgainsystems.com/wiki/display/GG15UG/MapReduce+with+Prime+Numbers`.

## References

[1] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics-Second Series*, 160(2):781, 2004.

[2] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.

[3] D. Angluin, J. Aspnes, and D. Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

[4] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *TAAS*, 3(4), 2008.

[5] M. Brown, S. Gilbert, N. Lynch, C. Newport, T. Nolte, and M. Spindel. The virtual node layer: a programming abstraction for wireless sensor networks. *SIGBED Rev.*, 4(3):7–12, 2007.

[6] I. Chatzigiannakis, O. Michail, and P. Spirakis. Decidable graph languages by mediated population protocols. Technical report, May 2009.

[7] I. Chatzigiannakis, O. Michail, and P. Spirakis. Mediated population protocols. In *36th International Colloquium on Automata, Languages and Programming (ICALP 2009)*, volume 5556 of *Lecture Notes in Computer Science*, pages 363–374. EATCS, Springer-Verlag Berlin Heidelberg, July 2009.

[8] I. Chatzigiannakis, O. Michail, and P. Spirakis. Recent advances in population protocols. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Springer-Verlag Berlin Heidelberg, August 2009.

[9] I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In G. Taubenfeld, editor, *DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 498–499. Springer, 2008.

[10] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[11] S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. Nolte. Timed virtual stationary automata for mobile networks. In J. H. Anderson, G. Prencipe, and R. Wattenhofer, editors, *OPODIS*, volume 3974 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2005.

[12] S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. L. Welch. Autonomous virtual mobile nodes. In *DIALM-POMC*, pages 62–69, 2005.

[13] S. Dolev, T. Herman, and L. Lahiani. Polygonal broadcast, secret maturity, and the firing sensors. *Ad Hoc Networks*, 4(4):447–486, 2006.

[14] S. Dolev, L. Lahiani, N. A. Lynch, and T. Nolte. Self-stabilizing mobile node location management and message routing. In T. Herman and S. Tixeuil, editors, *Self-Stabilizing Systems*, volume 3764 of *Lecture Notes in Computer Science*, pages 96–112. Springer, 2005.

[15] S. Dolev and N. Tzachar. Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. *Theor. Comput. Sci.*, 410(6-7):514–532, 2009.

[16] G. A. L. Paillard. A fully distributed prime numbers generation using the wheel sieve. In T. Fahringer and M. H. Hamza, editors, *Parallel and Distributed Computing and Networks*, pages 651–656. IASTED/ACTA Press, 2005.

[17] C. Pomerance. The quadratic sieve factoring algorithm. In *EUROCRYPT*, pages 169–182, 1984.

[18] C. Pomerance. A tale of two sieves. *Biscuits of Number Theory*, page 85, 2008.

[19] M. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.