Membrame dissolution in distributed architectures of P-Systems

MIGUEL ANGEL PEÑA
Universidad Politécnica de Madrid
Facultad de Informática
28660 Madrid
SPAIN
m.pena@upm.es

GINES BRAVO, LUIS FERNANDO DE MINGO
Universidad Politécnica de Madrid
Escuela Universitaria de Informática
Crta. De Valencia km. 7, 28031 Madrid
SPAIN
{gines,lfmingo}@eui.upm.es

Abstract: The goal of this paper is twofold. Firstly, to survey in a systematic and uniform way the main results regarding the way membranes can be placed on processors in order to get a software/hardware simulation of P-Systems in a distributed environment. Secondly, we improve some results about the membrane dissolution problem, prove that it is connected, and discuss the possibility of simulating this property in the distributed model. All this yields an improvement in the system parallelism implementation since it gets an increment of the parallelism of the external communication among processors. Also, the number of processors grows in such a way that is notorious the increment of the parallelism in the application of the evolution rules and the internal communicationsstudy because it gets an increment of the parallelism in the application of the evolution rules and the internal communications. Proposed ideas improve previous architectures to tackle the communication bottleneck problem, such as reduction of the total time of an evolution step, increase of the number of membranes that could run on a processor and reduction of the number of processors.

Key–Words: Membrane Computing, P-Systems Architectures, Distributed Communication, Membrame Dissolution

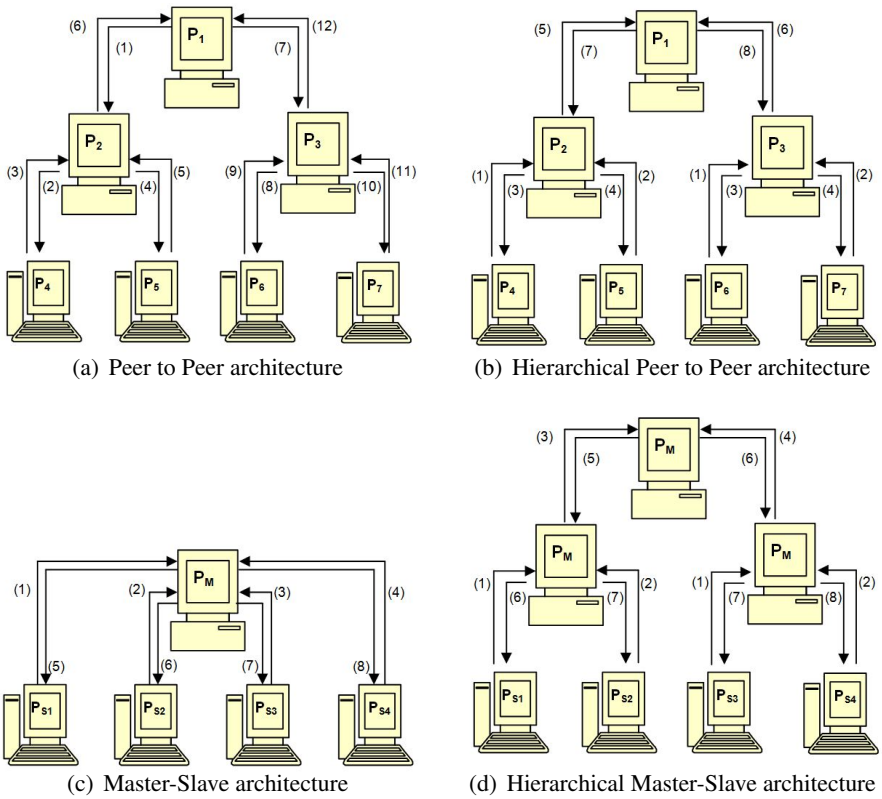
1 Preliminaries

Natural sciences, and especially biology, represented a rich source of modelling paradigms. Well-defined areas of artificial intelligence (genetic algorithms, neural networks), mathematics, and theoretical computer science (L systems, DNA computing) are massively influenced by the behaviour of various biological entities and phenomena. In the last decades or so, new emerging fields of so-called natural computing identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature, as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena.

Membrane computing, inspired in *"basic features of biological membranes"*, was introduced by Gheorge Paun [10] to solve *NP*-Complete problems in polynomial time. As original model –Transition P System– as remaining models emerging from its; they are an abstract representation of hierarchical structure and non-deterministic behavior of biological membranes. A membrane is a region compounds by other membranes and chemicals (objects) that uses chemical reactions (evolution rules) generated another chemicals. Each membrane has a permeability capacity that enable chemicals (objects) to move between membranes (communication). Chemicals reactions produced in membranes can dissolve it. This process implies that contained object and membranes to become part of parent membrane.

In base to this behaviour, P-System are systems that can be executed *on-line* that is, in vitro or simulated, using hardware implementations (Petreska [11], Fernandez [6] or Martinez [9]), using software simulations(Suzuki [12] o Arroyo [1]) or even in a real cluster of processors (Ciobanu [5] o Syropoulos [13]). Currently, researchers focused on simulations by distributed software, to alleviate the sequential nature of processors, to obtain lower running time. Must be distinguish two main steps in this system evolution: applying rules (with proper selection of it), and object communication between membranes. To apply rules, Frutos [7] proposed to create decision trees to determine possible rules to apply according to membrane context, and Gil [8] proposed algorithms to distribute objects among rules and its application. Objects communication between membranes depends on used distributed architecture. Since Ciobanu detected that network congestion produce higher response time, future studies have focused on searching architectures to eliminate network collision.

ISBN: 978-960-474-281-3



(a) Peer to Peer architecture (b) Hierarchical Peer to Peer architecture

(c) Master-Slave architecture (d) Hierarchical Master-Slave architecture

Figure 1: Different distributed P-System architectures including the object communication order.

2 P-Systems Architectures

The first distributed architecture to implement P-Systems that eliminate collisions is proposed like *"partially parallel evolution with partially parallel communication"* [14]; namely Peer-to-Peer ($P2P$) as opposed to other architectures. In this architecture (Figure 1(a)), processors are connected with a tree topology following the inner scheme in P-System structure: k membranes are placed in every processor P and sequentially executed applying rules. On communication step, following a post order way of processor tree (Figure 1(a), proxies situated on each processor send generated objects in its membranes, using only one message, to proxies of processors that are target membranes. Total time used in each evolution of P-System is:

$$T = KsT_{apl} + 2s(P-1)sT_{com} \qquad (1)$$

Based on the same idea and same tree topology, Bravo in [2] adds parallelism to transmission of objects, so simultaneously multiple processors can be communicating with others processors with no collision at all (Figure 1(b)). This architecture cited like Hierarchical Peer-to-Peer (H$P2P$) reduces, in a notorious way, time used in evolution steps:

$$T = \frac{M(A-1)T_{apl}}{A^L - 1} + 2sT_{com}sA(L-1) \qquad (2)$$

Where A is amplitude or processor number of children and L is number of processors tree levels, taking root like L=1.

Third architecture reduces time and eliminates complicated restrictions of tree topology of previous models, it is Master-Slave architecture (MS) [3]. With this architecture, also based on unique simultaneously communication between proxies (Figure 1(c)), can make the communication step while other processors are in their communication step. Time used by every evolution phase is:

$$T = ksT_{apl} + T_{com}(P_s + 1) \qquad (3)$$

Communications on Master-Slave architecture are sequentials and it produces that time of this step is linear. To address these issues arises Hierarchical Master-Slave architecture (HMS) [4] groups advantages of $HP2P$ architecture with MS architecture (Figure 1(d)), obtaining a time of:

$$T = \frac{MsT_{apl}}{A^{L-1}} + T_{com}(2L + A + A^{L-2} - 4) \qquad (4)$$

3 Behavior of distributed architectures

All this architectures are designed to reduce P-System running time, considering two execution steps of it. In addition to this stages, in different types of P-System may exists membrane dissolution or membrane division. There are to make changes to include these new steps in these architectures. $P2P$ and MS architectures may work with only the addition of a improvement proxy that permit dissolution. $HP2P$ and HMS architectures have to modify message flow to permit, through intermediate processors, communications among nodes even though it is not connected directly. This new communication sequence are shown in figures 2(a) (to H$P2P$) and 2(b) (to HMS).

$HP2P$ architecture starts its communication on root processor, which communicates with its children sequentially. Each one of these can starts communication with its children, when received father communication (root). In the same way, children communicate with it corresponding children. Reverse process, from leaves to root, is similar considering the restrictions that one processor only can have one communication to avoid collisions. When root received the last communication from it child, starts next evolution on application phase.

HMS architecture starts with communication with all slaves processors. Communications of all slaves are made in sequential, because there are many slaves processor connected to same master. But, two slaves of different master can starts to communicate in parallel, because there are not collisions. With the same idea, master can communicate with its masters-father, and so, to reach master-root. When it have received from whole its masters-children, made its process and sequentially communicate to whole masters-children. Each one of them, after receive the communication, it can communicate with its children (similarly to make to $HP2P$ architecture). When a master has terminated to communicate with its slaves, these can execute the application step of its next evolution phase, and it starts communication step when finished application step, avoiding collisions.

Except for the first evolution of P System, HMS and $HP2P$ architectures require the same time to communicate (as shown in figures 2(a) and 2(b)). Knowing that communication time between two processors (only one way) is T_{com}, total communication time for each evolution of the P System is:

$$T = T_{com}(AL + L - 2) \tag{5}$$

In both architectures, communications number behaves same, but its application time is different, because in $HP2P$ architecture exists membranes in every processor; and in HMS only exists in processors slaves. Existing k membranes on each processor, to take T_{apl} to apply, application time would be:

$$T = kT_{apl} \tag{6}$$

On $HP2P$ architecture there will be P processors:

$$P = \frac{A^L - 1}{A - 1} \tag{7}$$

On HMS architecture there will be the same number of total processors, of whom, processors slaves and masters, will be:

$$P_s = A^{L-1} \tag{8}$$

$$P_m = \frac{A^{L-1} - 1}{A - 1} \tag{9}$$

Knowing total number of membranes M is results of multiply membranes of each processor by the processors number that contain membranes, total times of each execution step to $HP2P$ and HMS respectively are:

$$T_{HP2P} = \frac{T_{apl}M(A-1)}{A^L - 1} + (LA + L - 2)T_{com} \tag{10}$$

$$T_{HMS} = \frac{MT_{apl}}{A^{L-1}} + (LA + L - 2)T_{com} \tag{11}$$

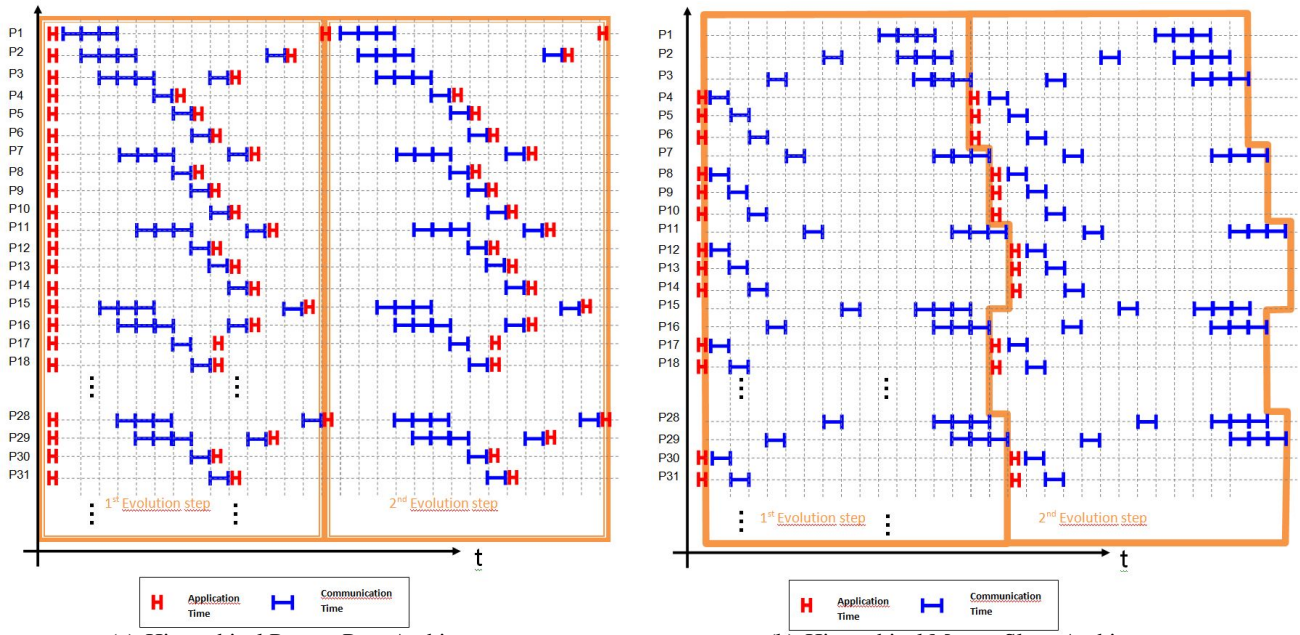
Knowing the time lasting each evolution step, it would be determined what is the optimum value of A and L to obtain minimum time. Also knowing that A and L must be integers. Figures 3(a) and 3(b) shown the better combination of A and L to be the minimum time, according relationship between T_{apl} and T_{com}, and number of membranes.

4 Results and Future Work

Entering membrane dissolution and membrane division step modifies the existing architectures behavior. The communications order of this architecture must also make these changes. Hierarchical Peer to Peer Architecture and Hierarchicla Master-Slave acquired like behavior. However, the total time and the distribution varies because in $HP2P$ are membranes on all processors and HMS only on slaves. Moreover, this change reduces the execution time.

To achieve a minimum time, should make other distributions different than when they took into account the dissolution of membranes. With this new configuration the number of children of a processor increases to 3 with less than 1000 membranes. The

ISBN: 978-960-474-281-3



(a) Hierarchical Peer to Peer Architecture

(b) Hierarchical Master-Slave Architecture

Figure 2: Chronogram of different P-System architectures with 4 depth levels and amplitude equal to 3.

optimal configuration for the number of children and the height of the tree of processors, depending on the number of membranes and the ratio of the time it takes to apply membranes and that take in communicating processors.

References:

[1] F. Arroyo, C. Luengo, A.V. Baranda and L.F. de Mingo, A software simulation of transition P System in Haskell,*Lecture Notes in Computer Science 2597,* Springer-Verlag, Berlin 2003, pp. 19-32.

[2] G. Bravo, L. Fernandez, F. Arroyo and J.A. Frutos, A Hierarchical Architecture with parallel communication for implementing P Systems, *Information Theories and Applications,* Vol 14. Varna 2007.

[3] G. Bravo, L. Fernandez, F. Arroyo and J. Tejedor, Master-Slave Distributed Architecture for Membrane System Implementation, *8th WSEAS Int. Conf. on Evolutionary Computing (EC'07),* June 2007, Vancouver, Canada.

[4] G. Bravo, L. Fernandez, F. Arroyo and M.A. Peña, Hierarchical Master-Slave Architecture for Membrane Systems Implementation, *Proceeding of the 13th Int. Symposium on Artificial Life and Robotics,* Jan 31-Feb 2, Beppu, Oita, Japan.

[5] G. Ciobanu and W. Guo, P System Running on a Cluster of Computers, *Workshop on Membrane Computing, LNCS 2933,* 2004, pp. 123-2004.

[6] L. Fernandez, V.J. Martinez, F. Arroyo and L.F. Mingo, A Hardware Circuit for Selecting Active Rules in Transition P Systems, *First International Workshop on Theroy and Application of P System,* September 2005, pp. 45-48.

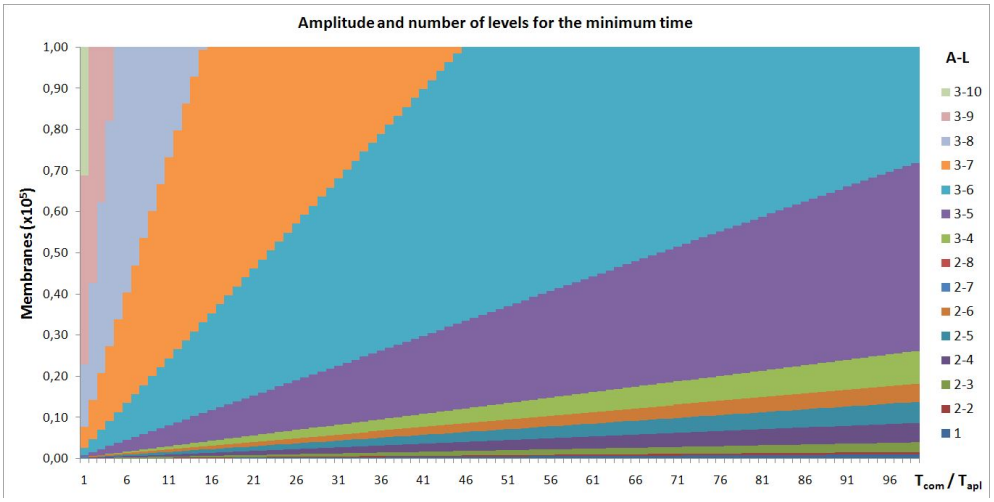
[7] J.A. de Frutos, L. Fernandez and F. Arroyo (2009), Decision Trees for Obtaining Active Rules in Transition P System,*Proceedings of the Tenth Workshop on Membrane Computing,* August 24-27, Romania.

[8] F.J. Gil, J.A. Tejedor and L. Fernandez, Fast Linear Algorithm for active rules application in Transition P Systems, *Sixth International Conference Information Research and Applications,* Varna 2008.

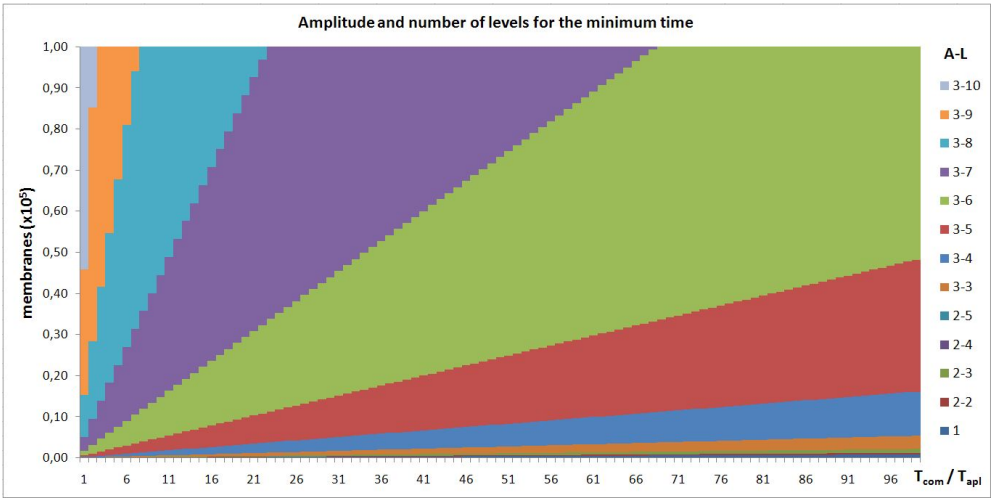
[9] V. Martinez, L. Fernandez, F. Arroyo and A. Gutierrez, Implementation HW of an Enclosed Algorithm to Rules Applications in a Transition P System,*8th International Symposium on Syumbolic and Numeric Algorithms for Scientific Computing,* 2006.

[10] G. Paun, Computing with membranes, *Journal of Computer and System Sciences,* 1(61). 2000, pp. 108-143.

[11] B. Petreska and C. Teuscher, A hardware membrane system,*Preproceedings of the Workshop on Membrane Computing,* 2003, pp. 242-255.



(a) $HP2P$ Architecture



(b) HMS Architecture

Figure 3: Amplitude and depth for the minimum time in P-Systems Architectures.

ISBN: 978-960-474-281-3

[12] Y. Suzuki and H. Tamaka, On a Lisp implementation of a class of P Systems,*Romanian Jounal of Information Science and Technology,* 2000, pp. 173-186.

[13] A. Syropoulos, E.G. Mamatas, P.C. Allilomes and K.T. Sotiriades, A distributed simulation of P Systems,*Preproceedings of the Workshop on Membrane Computing, 2003,* pp. 455-460.

[14] J. Tejedor, L. Fernandez, F. Arroyo and G. Bravo, An architecture for attacking the bottleneck communication in P systems, *Proceedings of the 12th Int. Symposium on Artificial Life and Robotics,* 2007, pp. 500-505.