An Architecture for Attacking the Bottleneck Communication in P Systems

J. A. Tejedor
Natural Computing Group
Universidad Politécnica de Madrid
Madrid, Spain

L. Fernández
Natural Computing Group
Universidad Politécnica de Madrid
Madrid, Spain

F. Arroyo
Natural Computing Group
Universidad Politécnica de Madrid
Madrid, Spain

G. Bravo
Natural Computing Group
Universidad Politécnica de Madrid
Madrid, Spain

Abstract

The distributed implementation of P system on a cluster of processors has met with the bottleneck communications problem. When the number of membranes grows in the system, the network gets congested and the times to execute an evolution step degrades.

In this paper, we suggest a sotfware architecture denominated "partially parallel evolution with partially parallel" communication where some membranes are located in each processor, proxys are used to communicate with membranes located in different processors and a policy of access control to the network communications is mandatory. With all this, we get a certain parallelism in the system and an acceptable functioning in the communications. In addition to this, it establishes a series of equations that allows us to determine in the architecture the optimum number of processors needed, the required time to execute an evolution step, the number of membranes to be located in each processor and the conditions to determine when it is best to use the distributed solution or the sequential one.

keywords: architecture, bottleneck, communication, P systems

1 Introduction

The transition P systems were presented by Gheorghe Păun in 1998 [1] who based his work on basic features of biological membranes. A membrane defines a region where a series of chemical elements (multisets) may experience a series of chemical reactions (evolution rules) and produce other elements. Inside the region limited by a membrane may be, at the same time, other membranes creating a complex hierarchical structure that may be represented by a tree. The products generated by the chemical reactions may stay in the same region or travel to the container region or to the regions contained by a membrane. As a result of such reaction, a membrane may dissolve itself (its chemical elements transfer to the container membrane) or inhibit itself (the membrane becomes impermeable and does not allow any object to pass).

The membranes systems are dynamic as the chemical reactions produce elements that cross the frontiers of the membranes to travel to other regions in order to produce new reactions. This dynamic behaviour can be sequenced in a series of evolution steps between one and another configuration system that will be determined by the membrane structure and multisets present inside membranes. In the transition P systems formal model two phases are distinguished in each evolution step: rules application and communication. In the rules application phase inside each membrane its rules are applied to the multisets in parallel. Once the previously described phase has concluded, the communication phase begins and the generated multisets travel towards the target membranes. These systems perform a computation through transition between two consecutives configurations, transforming into computational device with the same capacities that Turing machines.

The power of this computation model lies in the fact that the process is massively parallel in the rules application phase as well as in the objects communication phase. The challenge for the researchers is

to get hardware or software implementations of P systems with a high parallelism degree.

A natural and intuitive implementation of P systems in electronic devices is carried out based on the following:

1. To locate in each processor a membrane where the evolution rules are applied parallelly on their multiset.

2. To carry out the communication between 2 membranes (2 processors) using 4 interfaces and 2 communications buses. The data travel in each direction (father-son or son-father) using a communication interface in the outgoing processor and another one in the incoming processor that are connected by a data bus.

The main problem of this implementation model is that it is unfeasible. Nowadays technologies do not allow for a processor to have as many communication interfaces as membranes are connected to it.

The aim of this work is to establish a communications architecture that will adapt to the special features of P systems. For this, it is structured in the following way: in the first place, the related works are enumerated analyzing the proposed architectures, next a communication architecture model is introduced stating its economical and computational cost as well as its viability, to continue with a more detailed analysis of the model, to end describing the conclusions obtained.

2 Related Works

The implementation of P system in digital hardware device is being carried out from the point of view of Hardware as well as Software. Most of the solutions have been focused, mainly, in the first phase of the P system evolution describing digital circuits or software architectures/designs that have allowed the application of the defined evolution rules inside the membranes. The phase of multisets membranes communication has not been contemplated or it has simply been performed by shared memory, except Syropoulos [2] and Ciobanu [3] that in their distributed implementations of P systems use Java Remote Method Invocation (RMI) and the Message Passing Interface (MPI) respectively, on a cluster of PC connected by Ethernet. These last authors do not carry out a detailed analysis of the importance of the time used during communication phase in the total time of P system evolution, although Ciobanu affirms that "the response time of the program has been acceptable. There are however executions that could take a rather long time due to unexpected network congestion" [3].

A model of impementacion of P system simplifying and generalizing the ideas of the works of Syropoulos and Ciobanu would be the following one:

1. In each processor a membrane is located where its rules will be applied. A processor can be a digital circuit implemented by Field-Programmable Gate Arrays (FPGAs), a microcontroller or a microprocessor.

2. All processors are connected to a common bus through a communication interface governed by a protocol.

In this model, all the membranes apply its rules in parallel for later communicating among them. Due to the fact that the communication line is common to all of them, at a particular time there will only be a membrane or processor communicating. Then, the communication becomes sequentially. The problem is more complex as Ciobanu advices [3], because if no special measures are taken, there may be more than one processor trying to communicate at a particular time and collisions will produce that will delay the whole communication process.

Assuming that no collisions happen in the network and taking into consideration that the information transfer between 2 membranes is made in both directions (father-son/son-father), the total number of communications made up during a transition step is $2(M-1)$. Therefore, the total time to perform the evolution step is:

$$T = T_{apl} + 2(M-1)T_{com} \quad (1)$$

where M is the number of membranes of the P system, T_{apl} is the maximum time used by the slowest membrane in applying its rules in the whole system and T_{com} is the maximum value of all times of communication between 2 nodes. Meaning that the time T is linear dependent on the number of membranes that the P system has.

The system throughput (processors and communications) can be expressed in accordance with the following:

$$Th_{evo} = \frac{T_{apl}}{T_{apl} + 2(M-1)T_{com}} \quad (2)$$

$$Th_{com} = \frac{2(M-1)T_{com}}{T_{apl} + 2(M-1)T_{com}} \quad (3)$$

So if the number of membranes grows, the processors throughput will approach zero, while the communications will practically be working 100% of the time.

Considering the system costs, this model is expensive because it needs as many processors and communication interfaces as membranes in the system:

$$C = MC_{pro} + MC_{com} \quad (4)$$

The analysis of this model makes us reach the following conclusions:

- The time used in a step of P system evolution is huge and it grows lineally with the number of membranes.
- The system cost is very high with the processors throughput practically nil.
- Considering that T_{apl} is normaly smaller than T_{com}, it complies with:

$$MT_{apl} < T_{apl} + 2(M-1)T_{com} \quad (5)$$

The time used in a step of P system evolution is worse than the one obtained using a single processor. Consequently, it is useless to choose this software architecture for the P systems implementation.

Therefore, it is necessary to carry out a study of the communications problem in a processors distributed architecture in order to determine the viability implementation of a P system.

3 Partially parallel evolution with partially parallel communication Model

The communication model that is being introduced in this section allows a certain parallelism degree in the rules application phase, as well as in a P system communication phase. In this way, the system cost decreases and the acceptable time used in a P system evolution step is obtained. In short, this new model is denominated "partially parallel evolution with partially parallel communication" and it is based on the following:

1. **Membranes Distribution**: In each processor, K membranes are located that will evolve, at worst, sequentially. The value of K is determined by the relation between the number of membranes M and processors P, where $K \geq 1$.

   The benefit obtained is that the number of the external communications decreases. The total number of communications splits in two classes: a group of internal communications for pairs of membranes located in the same processor and another group of external communications to interchange information among pairs of membranes located in different processors. Therefore, the number of external communications against the previous model will always be smaller. Moreover, this is an important fact because the run time to carry out the internal comunications will be negligible.

   For example, the 22 external communications performed by an architecture with a membrane located in each processor (Figure 1) have been reduced to 10 in the architecture that has located 3 membranes in 4 processors (Figure 2).

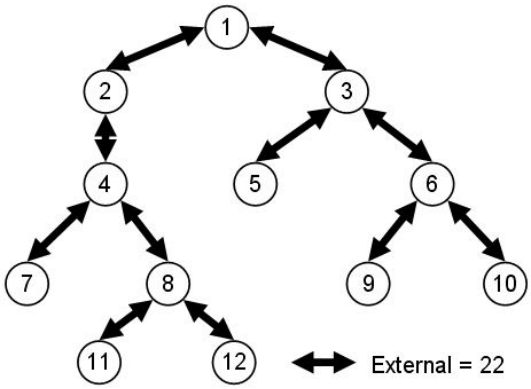

Figure 1: P system communications

2. **Proxy for processor**: When a membrane wants to communicate with another one located at a different processor, the first one uses a proxy (programs or device located in the processor that carries out an action in representation of another), instead of doing it directly. Therefore, the communications that use the common line (external communications to the processor) are carried out between proxys, not between membranes. This intermediate element located between the bus and the membranes concentrates the information in two stages:

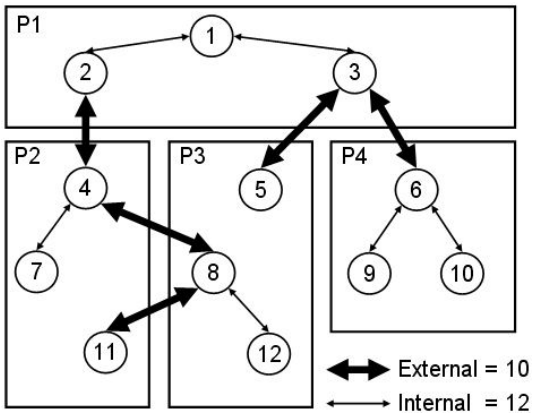

Figure 2: Communications with membranes distribution

(a) N multisets of N membranes located in a processor that has a common father membrane in another processor, becoming integrated in a single multiset that is the one that will be sent.

(b) The S communication packet of L length necessary to communicate between S pairs of membranes located in 2 different processors are reduced to one single packet of $S.L$ length.

The benefit of using proxys in the communication among membranes against direct communication is double:

(a) Due to the first stage previously described, the amount of information sent is smaller. This is produced by the fact that the N packet necessary to communicate N membranes with the same father, are transformed into a single packet of the length of a single multiset.

(b) Due to the second stage, the number of external communications is smaller although packets are bigger. But, considering that the communication protocols penalize the transmission of small packets because to the data encapsulated processes and to the time safety intervals between future transmission, it is better to send one packet of length equal $S.L$ than S packets of length equal L.

Figure 3 shows that if proxys are introduced in the processors, then the number of external communications are reduced to 8.

3. **Tree topology of processors**: In graph theory it is established that $P-1$ connections is the minimum number required to interconnect a connected graph of P processors. This restriction imposes on the graph a tree topology. The benefit obtained with the tree topology of processor is that it minimizes the total number of external communications made as the proxys interchange information only with its direct predecessor and its direct successors, and therefore the total number of external communications in each evolution step is $2(P-1)$.

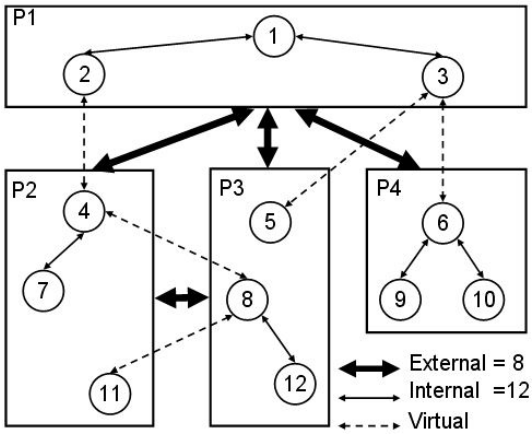

Figure 3: Communications with a proxy for processor

Figure 4 shows that external communications are reduced to 6 when a tree topology of procesors is used to connect them.

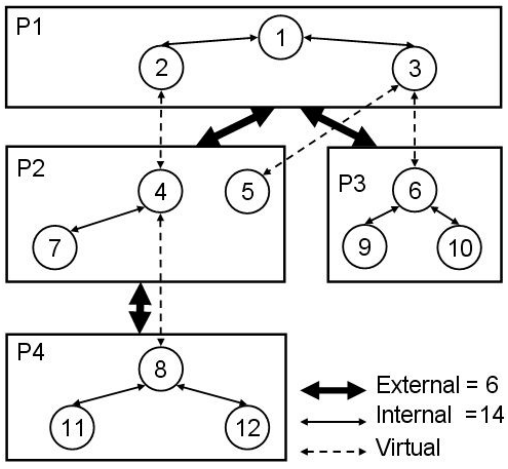

Figure 4: Communications using a tree topology

4. **Token passing in the communication:** No proxy can start a communication until it is invited to do it by means of token passing. This token travels through a depth first serarch sequence in the topology of processors tree. In this way, when a X proxy receives a communication from its father proxy acquires the token and then sends information to its first child proxy (C1) passing the token and keeps waiting. When C1 sends information in answer to X, the later acquires the token again and sends information to its second son proxy (C2) passing the token and keeps waiting, and so on until all father-childrens communications have been carried out. Finally, X proxy sends its answer to its father returning the token. The whole process starts with the root proxy of processors hierarchy.

   This communication policy prevents that more than one proxy are trying to transmit information at the same time. Therefore, there are no collisions and no congestion in the line. Figure 5 shows the communication sequence of the four processors proposed in the sample.

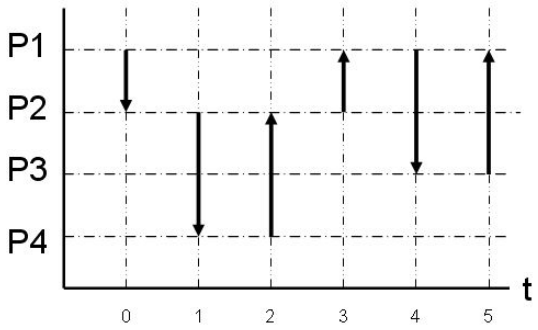

Figure 5: Communication sequence

4 Detailed analysis of the partially parallel evolution with partially parallel communication model

In this software architecture K membranes have been located in each processor. At the worst, the application of the rules in each one of these membranes will be made sequentially in each processor. Therefore, the run time to carry out the application of the rules of M membranes will be:

$$KT_{apl}$$

Due to the establishment of tree topology of processors, the number of external communications in each evolution step of the P system will be:

$$2(P-1)$$

Therefore, the required time to perform a complete evolution step will be:

$$T = KT_{apl} + 2(P-1)T_{com\_pro} \quad (6)$$

Being T_{com_pro} the maximum time required to communicate 2 proxys using the common bus. T_{com_pro} value will depend on the topology of P system and also the distribution that has been made of M membranes in P processors. In the most favourable case, when a unique membrane located in a processor has to communicate with a unique membrane located in another processor, T_{com_pro} is similar to T_{com}. The worst case takes place when K membranes located in a processor send information to other K membranes located in a different processor. However, keeping in mind that the protocol penalizes the short packets (there is not much difference sending 10 or 10000bytes through TCP/IP according to the experimental data obtained), that the encapsulation processes take its time and that the information can be compressed before sending it, we can assume that T_{com_pro} is similar to the product of a constant by T_{com}, being this constant much smaller that K when K is big. Therefore,

$$T_{com\_pro} = cT_{com}\ where\ c >= 1 << K \quad (7)$$

Once it is known the required time to perform an evolution step, we can determine the number of membranes that should be located in each processor in order to minimize the time:

$$K_{opt} = \left\lceil \sqrt{\frac{2cMT_{com}}{T_{apl}}} \right\rceil where\ 1 <= K_{opt} <= M \quad (8)$$

This K_{opt} value allows to calculate the number of processors necessary to run the P system minimizing the necessary time to carry out an evolution step.

$$P_{opt} = \left\lceil \sqrt{\frac{M.T_{apl}}{2cT_{com}}} \right\rceil where\ 1 <= P_{opt} <= M \quad (9)$$

From the values K_{opt} and P_{opt} the minimum time required to perform an evolution step is:

$$T_{min} = 2\sqrt{2cMT_{apl}T_{com}} - 2cT_{com} \quad (10)$$

Therefore, the system evolution time is obtained by adding twice the square root of the result of multiplying the number of membranes M by T_{apl} and by T_{com}.

The processors and the communications throughput is calculated as follows:

$$Th_{pro} = \frac{\sqrt{2cMT_{apl}T_{com}}}{2\sqrt{2cMT_{apl}T_{com}} - 2cT_{com}} \quad (11)$$

$$Th_{com} = \frac{\sqrt{2cMT_{apl}T_{com}} - 2cT_{com}}{2\sqrt{2cMT_{apl}T_{com}} - 2cT_{com}} \quad (12)$$

If we disregard the $2cT_{com}$ term, the value got in both cases is 0.5, therefore, a more balanced system has been obtained (50% working the processors, 50% working the communications) than the one obtained with the other software architectures.

With regards to the cost of this architecture, we can assure that it is moderate against the other architectures proposed, as P processors are needed with P communication interfaces and the value of P is around the square root of the number of membranes M.

$$C = PC_{pro} + PC_{com} \quad (13)$$

About the main factors that influence in the time used in a step of P system evolution in this software architecture (T_{apl} and T_{com}) we really can influence only on the first one in order to improve this time. The software engineers can make that the K membranes of a processor apply faster the evolution rules, thus developing faster sequential or parallel algorithms. Nevertheless, it is difficult to get faster communicate among processors as it is necessary to invest many resources that are only within reach of the telecommunications industry. If it is possible to make that T_{apl} be N faster times the values of K_{opt}, T_{opt} and T_{min} will be:

$$K_{opt} = \left\lceil \sqrt{\frac{2cMNT_{com}}{T_{apl}}} \right\rceil where\, 1 <= K <= M \quad (14)$$

$$P_{opt} = \left\lceil \sqrt{\frac{M.T_{apl}}{2cNT_{com}}} \right\rceil where\, 1 <= P <= M \quad (15)$$

$$T_{min} = 2\sqrt{\frac{2cMT_{apl}T_{com}}{N}} - 2cT_{com} \quad (16)$$

Therefore, the number of membranes that would be runned in a processor would be multiplied by $\sqrt{N}$, the number of required processors would be divided by the same factor and the time required to perform an evolution step would improve approximately with the same factor $\sqrt{N}$.

Finally it is important to know, when a distributed architecture is better than a momoprocessor architecture to perform a computation. Therefore we will have to determine under which conditions the following is fullfilled:

$$2\sqrt{2cMT_{apl}T_{com}} - 2cT_{com} < MT_{apl} \quad (17)$$

Then, the several processors solution is better when:

$$M > \frac{2cT_{com}}{T_{apl}} \quad (18)$$

5 Conclusions

In this paper a communications architecture to implement P system has been introduced. This architecture is based on the location of several membranes in the same processors, the use of proxys for communicating processors placed in a tree topology and token passing in the communication. This solution avoids communication collisions, reduces the number and length of the external communications. All this, allows to obtain a better step evolution time than in other suggested architectures congested quickly by the network collisions when the number of membranes grows. Also, our architecture is highly scaleable with moderate costs.

References

[1] Gh. Păun (2000), "Computing with membranes", Journal of Computer and System Sciences, 61, 1, 108-143

[2] A. Syropoulos, E.G. Mamatas, P.C. Allilomes et al (2003), "A distributed simulation of P systems", A. Alhazov, C. Martin-Vide and Gh. Păun (Editors): Preproceedings of the Workshop on Membrane Computing; Tarragona, July 17-22, 455-460

[3] G.Ciobanu, W.Guo (2004), "P Systems Running on a Cluster of Computers". Workshop on Membrane Computing (Gh. Păun, G. Rozenberg, A. Salomaa Eds.), LNCS 2933, Springer, 123-139.